

REPORT DOCUMENTATION PAGE

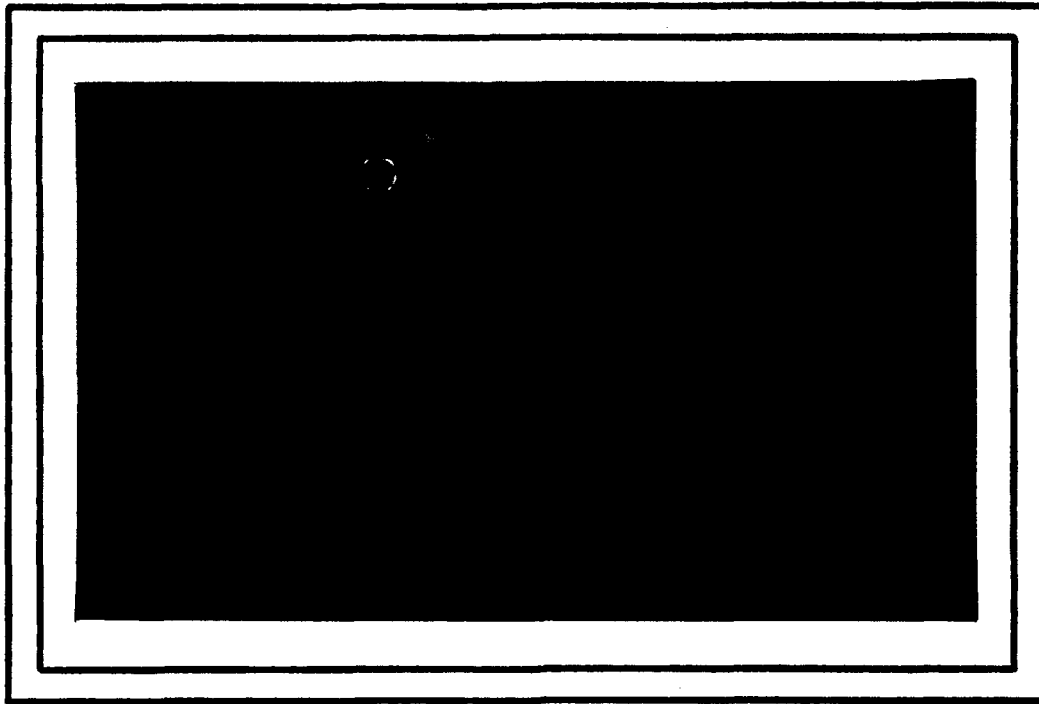
Form Approved •
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE Mar 90	3. REPORT TYPE AND DATES COVERED Technical	
4. TITLE AND SUBTITLE Generalized Disjunctive Well-Founded Semantics for Logic Programs			5. FUNDING NUMBERS DAAG29-85-K-0177	
6. AUTHOR(S) Chitta Baral, Jorge Lobo, Jack Minker			<div style="text-align: center;"> DTIC SELECTED FEB 26 1991 S B D </div>	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Maryland College Park, MD 20742				
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Research Office P. O. Box 12211 Research Triangle Park, NC 27709-2211			10. SPONSORING/MONITORING AGENCY REPORT NUMBER ARO 22648.19-EL	
11. SUPPLEMENTARY NOTES The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) Generalized disjunctive well-founded semantics (GDWFS) is an extension of generalized well-founded semantics (GWFS) of Baral, Lobo and Minker, to disjunctive logic programs. We describe fixpoint, model theoretic and procedural semantics and show their equivalence. The fixpoint semantics is similar to the fixpoint semantics of GWFS, except that it iterates over states (a pair of sets; one a set of disjunctions of atoms and the other a pair of conjunctions of atoms), rather than partial interpretations. The model theoretic semantics is based on a dynamic stratification of the program. The procedural semantics is based on SLIS refutations, +trees and SLISNF trees. We compare the GDWFS with the strong well-founded semantics of Ross and the stationary model semantics of Przymusinski.				
14. SUBJECT TERMS			15. NUMBER OF PAGES	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

AD-A232 064

ALL COPY



**COMPUTER SCIENCE
TECHNICAL REPORT SERIES**



**UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND**

20742

UMIACS-TR-90-39
CS-TR -2436

March 1990

Generalized Disjunctive Well-founded Semantics for Logic Programs

Chitta Baral¹, Jorge Lobo¹, and Jack Minker^{1,2}

Institute for Advanced Computer Studies²
and

Department of Computer Science¹
University of Maryland
College Park, MD 20742

Abstract

Generalized disjunctive well-founded semantics (GDWFS) is an extension of generalized well-founded semantics (GWFS) of Baral, Lobo and Minker, to disjunctive logic programs. We describe fixpoint, model theoretic and procedural semantics and show their equivalence. The fixpoint semantics is similar to the fixpoint semantics of GWFS, except that it iterates over states (a pair of sets; one a set of disjunctions of atoms and the other a pair of conjunctions of atoms), rather than partial interpretations. The model theoretic semantics is based on a dynamic stratification of the program. The procedural semantics is based on SLIS refutations, +trees and SLISNF trees. We compare the GDWFS with the strong well-founded semantics of Ross and the stationary model semantics of Przymusiński.

1 Introduction and Motivation

Disjunctive logic programs are natural extensions of Horn logic programs to represent disjunctive and indefinite information elegantly. For example if we know that Fred is either a bird or a mammal, and both mammals and birds are living beings, then we should be able to conclude that Fred is a living being. If there is no reason for us to prefer between the possibilities of Fred being a bird or a mammal we can not represent this information as a general Horn program consisting of clauses $bird(Fred) \leftarrow \neg mammal(Fred)$ or $mammal(Fred) \leftarrow \neg bird(Fred)$. Hence a natural way is to represent this information as the disjunctive clause $bird(Fred) \vee mammal(Fred)$.


General disjunctive programs allow negation in their bodies. Formally, a general disjunctive logic program is a set of clauses of the form $A_1 \vee \dots \vee A_n \leftarrow L_1 \wedge \dots \wedge L_m$, where the A 's are atoms and the L 's are literals. Various semantics for general Horn logic programs have been suggested that can handle general logic programs that are not stratified. Van Gelder et al. [VRS88] introduced well-founded semantics for general Horn logic programs and Przymusinski [Prz89b] presented well-founded semantics for general Horn logic programs in terms of a three-valued logic. Gelfond and Lifschitz [GL88] described stable model semantics for general Horn logic programs. Baral, Lobo and Minker [BLM89] developed the generalized well-founded semantics for general Horn logic programs.

For disjunctive logic programs, Ross [Ros] developed the strong well-founded semantics and Przymusinski [Prz] developed the stationary model semantics. In this paper we develop the generalized disjunctive well-founded semantics for logic programs and compare it with the approaches by Ross and Przymusinski.

The ideas underlying the generalized disjunctive well-founded semantics can be motivated by the following example. Let P be the logic program:

$$\begin{aligned} a &\leftarrow \neg b, c, d \\ b &\leftarrow \neg a \\ c &\leftarrow \neg d \\ d &\leftarrow \neg c \end{aligned}$$

If we consider all minimal models of the program P : $\{c, a\}$, $\{c, b\}$, $\{d, a\}$ and $\{d, b\}$, the conjunction $c \wedge d$ can be seen to be false in every minimal model, even though both c and d

		<input checked="" type="checkbox"/> OK <input type="checkbox"/> <input type="checkbox"/>	
		<input type="checkbox"/>	
ty Codes		Avail and/or Special	
Dist A-1			

d are not individually false (or even true) in all minimal models of P . Because of this it is reasonable to assume the conjunction of c and d , $c \wedge d$, to be false. This makes the body of the only rule with a in its head false. Therefore, it is also reasonable to assume a to be false and consequently b to be true. In [BLM89], we developed the generalized well-founded semantics for general Horn programs which assigns the truth value undefined to b and to a . Although the generalized well-founded semantics extended the well-founded semantics we are not able to capture the above meaning, because there, we use partial interpretations to assign truth values. A partial interpretation I is a pair $\langle T, F \rangle$ where T is the set of atoms assumed to be true and F is the set of atoms assumed to be false. We were not able to capture the falsity of conjunctions like $c \wedge d$ using only I . In this paper instead of using a partial interpretation I , we use the concept of a state S , which is a pair $\langle T', F' \rangle$, where T' is a set of positive clauses and F' is a set of conjunctions of atoms, to extend the generalized well-founded semantics to disjunctive programs. The generalized disjunctive well-founded semantics for logic programs that we present handles both disjunctive programs and captures the above aspects when restricted to general Horn logic programs.

2 Definitions

A *general logic program* is a finite set of clauses of the form

$$A_1 \vee \dots \vee A_n \leftarrow L_1 \wedge \dots \wedge L_m$$

where $n \geq 1$, $m \geq 0$, the A s are atoms and the L s are literals (i.e. positive or negated atoms). A *general Horn clause* is a clause where $n = 1$. A *general Horn program* or a *normal program* is a general logic program that consists of general Horn clauses. A *general indefinite* or *general disjunctive clause* is one where $n \geq 2$. A general logic program is called *general disjunctive* if it contains a general disjunctive clause. We use the term *program* to refer to both logic programs and general logic programs.

The *Herbrand Universe* U_P of a program P , is the set of all ground terms that can be formed from the constants and function symbols in P (if there are no constants in P an arbitrary constant is placed in U_P). The *Herbrand Base* of a logic program P , $HB(P)$, is defined as the set of all ground atoms that can be formed by using predicate symbols from P with terms from the Herbrand Universe U_P as arguments [Llo84]. A *Herbrand interpretation*

I for P is a subset of the Herbrand Base of P , in which all atoms in I are assumed to be *true* while those not in I are assumed to be *false*. An *Herbrand model* of P is a Herbrand interpretation of P that makes all clauses in P true. A model M is a minimal model of a program P iff M is a model of P and no proper subset of M is a model of P .

3 Semantics for Disjunctive Logic Programs

In contrast to Horn programs, information that can be derived from disjunctive programs might not be atomic. Consider the disjunctive program $\{A \vee B \leftarrow C, C\}$. From this program we can deduce C and $A \vee B$, but we are not able to derive either A or B . As a consequence, any semantics that we intend to develop for disjunctive programs must handle indefinite information. Minker and Rajasekar [MR90] developed a declarative semantics for disjunctive programs P , based on the extended Herbrand base of the program, $EHB(P)$, the set of all positive clauses that can be formed using distinct atoms from the Herbrand base $HB(P)$. The semantics is described as the least fixpoint of the following operator:

Definition 3.1 [MR90] (Operator T_P^I)

Let S be a subset of $EHB(P)$, then

$$T_P^I(S) = \{ C \in EHB(P) \mid C' \leftarrow B_1, \dots, B_n \text{ is a ground instance of a clause in } P \text{ and} \\ \forall i, 1 \leq i \leq n, B_i \vee C_i \in S, \text{ where } C_i \text{ can be null, and} \\ C'' = C' \vee C_1 \vee \dots \vee C_n \text{ and } C \text{ is the smallest factor of } C'' \}.$$

□

The smallest factor of a ground clause C' is defined as the clause C such that C contains only distinct atoms and C logically implies C' and C' logically implies C . T_P^I is continuous and the least fixpoint $lfp(T_P^I)$ characterizes derivability from a disjunctive program P . Formally.

Theorem 1 [MR90] Given a program P ,

$$lfp(T_P^I) = \{C \mid C \text{ derivable from } P\}.$$

□

In addition, Minker [Min82] described how to derive negative information using the Generalized Closed World Assumption (GCWA). A ground atom A is assumed false by the GCWA if A is false in all minimal (Herbrand) models of P . Yahya and Henschen [YH85] developed a natural extension of the GCWA where they allow “indefinite” negative information to be deduced, i.e., we might assume that $A \wedge B$ is false ($\neg A \vee \neg B$ is true) without

necessarily knowing that A or B is false. This kind of assumption will be necessary to describe the semantics of programs like the one given in the introduction. Roughly speaking, this extension to the GCWA can be defined as follows: a conjunction of ground atoms can be assumed to be false if the conjunction is false in all minimal (Herbrand) models of P . In this paper we extend the semantics for general disjunctive programs described in [MR90] and [RM88] to include the extended GCWA. For this, we characterize the meaning of a general disjunctive program by a pair $\langle T, F \rangle$, where T is a set of disjunctions of atoms and F is a set of conjunctions of atoms. More formally:

Definition 3.2 *For a general disjunctive program P , the Conjunctive Herbrand Base of P , $CHB(P)$, is the set of all conjunctions of atoms that can be formed using distinct elements from the Herbrand base of P , $HB(P)$.*

Example 3.1 *For the program $P = \{p(x) \leftarrow q(x); q(a) \vee p(a)\}$, the conjunctive Herbrand base is: $CHB(P) = \{p(a), q(a), p(a) \wedge q(a)\}$.*

Definition 3.3 *A state S is a pair $\langle T, F \rangle$, where*

1. *T is a subset of $CHB(P)$ such that if $C \in T$ then $C' \in T$ for all clauses C' subsumed by C .*
2. *F is a subset of $CHB(P)$ such that if $C \in F$ then $C' \in F$ for all clauses C' such that $\neg C'$ is subsumed by $\neg C$.*

Example 3.2 *The following pair, $\langle T, F \rangle$, forms a state for the program P in Example 3.1:*

$$T = \{p(a), p(a) \vee q(a)\}.$$

$$F = \{q(a), p(a) \wedge q(a)\}.$$

We call any element in $CHB(P)$ a disjunct and any element in $CHB(P)$ a conjunct. For a set of formulas S , we denote $\neg S$ the set $\{\neg\psi : \psi \in S\}$. A state $S = \langle T, F \rangle$ is consistent if $T \cup \neg F$ is consistent. The state in Example 3.2 is a consistent state. States will play the role of partial interpretation in general disjunctive programs (see [BLM89]). We define the truth value of a sentence with respect to a state $S = \langle T, F \rangle$ as follows:

1. If $G \in T$ then $val_S(G) = \text{true}$.
2. If $G \in F$ then $val_S(G) = \text{false}$.
3. $val_S(\neg G) = \neg val_S(G)$.
4. If $val_S(G) = \text{true}$ and $val_S(H) = \text{true}$ then $val_S(G \wedge H) = \text{true}$.
5. If $val_S(G) = \text{true}$ or $val_S(H) = \text{true}$ then $val_S(G \vee H) = \text{true}$.
6. For a formula $G(x)$ with free variable x , $val_S(\forall x G(x)) = \text{true}$ if $val_S(G(t)) = \text{true}$ for all term t in U_P .
7. For a formula $G(x)$ with free variable x , $val_S(\forall x G(x)) = \text{false}$ if $val_S(G(t)) = \text{false}$ for some term t in U_P .
8. For a formula $G(x)$ with free variable x , $val_S(\exists x G(x)) = \text{false}$ if $val_S(G(t)) = \text{false}$ for all term t in U_P .
9. For a formula $G(x)$ with free variable x , $val_S(\exists x G(x)) = \text{true}$ if $val_S(G(t_1)) \vee \dots \vee val_S(G(t_n)) = \text{true}$ for some finite subset $\{t_1, \dots, t_n\}$ of terms in U_P .
10. For any other sentence G , $val_S(G) = \text{undefined}$.

We also define $\neg \text{true} = \text{false}$, $\neg \text{false} = \text{true}$ and $\neg \text{undefined} = \text{undefined}$. In the remainder of this paper we write t for true, f for false and u for unknown. We extend the operators \mathcal{T}_I and \mathcal{F}_I defined in [BLM89] to handle general disjunctive programs. As defined formally, below, the operator \mathcal{T}_S^D assigns to every set T of disjuncts a new set $\mathcal{T}_S^D(T)$ of disjuncts and the operator \mathcal{F}_S^D assigns to every set F of conjuncts a new set $\mathcal{F}_S^D(F)$ of conjuncts. The superscript D in, \mathcal{T}_S^D and \mathcal{F}_S^D denotes that we are dealing with Disjunctive logic programs. Intuitively, $\mathcal{T}_S^D(T)$ contains new positive disjunctive facts (i.e disjuncts not contained in S), whose truth can be derived in one step from the program P assuming that all facts in S hold and all disjuncts in T are true. $\mathcal{F}_S^D(F)$ contains new conjunctions of atoms (i.e. conjuncts not contained in S), whose falsity can be derived in one step from the program P assuming that formulas in S are true and all conjuncts in F are false.

Definition 3.4 Let $S = \langle T_S, F_S \rangle$ be a state, P be a general disjunctive program, T be a subset of $EHB(P)$ and F be a subset of $CHB(P)$. Let the B s and A be ground atoms, the C s and E be disjuncts and G be a conjunct in the following formulas:

$\mathcal{T}_S^D(T) = \{ C \in EHB(P) : \text{val}_S(C) \neq t, \text{val}_S(C) \neq f \text{ and } \exists C' \leftarrow B_1, \dots, B_m, \neg B_{m+1}, \dots, \neg B_n \text{ in } P \text{ and a ground substitution } \theta \text{ such that for all } i, 1 \leq i \leq m, (B_i \vee C_i)\theta \in T_S \text{ or } (B_i \vee C_i)\theta \in T, C_i\theta \text{ might be null, } \{B_{m+1}, \dots, B_n\} \subseteq F_S \text{ and the smallest factor of } (C' \vee C_1 \vee \dots \vee C_m)\theta \text{ subsumes } C. \}$

$\mathcal{F}_S^D(F) = \{ G \in CHB(P) : \text{val}_S(G) \neq f, \text{val}_S(G) \neq t \text{ and for all ground instances of rules } A \vee E \leftarrow B_1, \dots, B_m, \neg B_{m+1}, \dots, \neg B_n \text{ in } P, \neg A \text{ subsumes } \neg G \text{ and at least one of the following four cases holds:}$

1. $B_{m+1} \vee \dots \vee B_n \in T_S.$
2. $B_1 \wedge \dots \wedge B_m \in F_S.$
3. $B_1 \wedge \dots \wedge B_m \in F.$
4. $E \in T_S. \}$

□

The operator \mathcal{T}_S^D is similar to the operator T^S defined by Minker and Rajasekar for the fixpoint semantics of disjunctive programs [MR90]. For the operator $\mathcal{F}_S^D(F)$, we can assume A (and all conjuncts that contain A) to be false if in all rules where A occurs in the head, either the body is false with respect to S and F (by assumption, there is no way that we can deduce a disjunction that contains A using this rule), or the disjunct E associated with A in the head of the rule is true with respect to S (A can be assumed to be false by the GCWA).

3.1 The Direct Extension of Well-founded Semantics is Not Sufficiently Expressive

For general Horn programs, well-founded semantics [VRS88, Prz89b] is given as a fixpoint of an operator \mathcal{I} , defined as $\mathcal{I}(I) = I \cup \langle T_I; F_I \rangle$, where I is a partial model and the definitions of T_I and F_I are analogous to \mathcal{T}_S^D and \mathcal{F}_S^D in the atomic case, i.e. given a partial interpretation I and a program P , T_I and F_I are defined as iterations of the following operators:

$\mathcal{T}_I(T) = \{ A : \text{val}_I(A) \neq t, \text{val}_I(A) \neq f \text{ and } \exists B \leftarrow L_1, \dots, L_m \text{ in } P, \text{ such that for all } i,$

$1 \leq i \leq m$, $L_i\theta$ is true in I or $L_i\theta \in T$, and $A = B\theta$ for a ground substitution θ).

$\mathcal{F}_I(F) = \{ A : \text{val}_I(A) \neq f, \text{val}_I(A) \neq t \text{ and for all rules of type } B \leftarrow L_1, \dots, L_m \text{ and ground substitutions } \theta \text{ where } A = B\theta, \text{ at least one of the following two cases hold:}$

1. $\exists i : i \leq m \text{ and } L_i\theta \text{ is false in } I.$
2. $\exists i : i \leq m \text{ and } L_i\theta \in F\}.$

Then,

$$\begin{aligned} T_I^{\uparrow 0} &= \emptyset \\ T_I^{\uparrow n+1} &= \mathcal{T}_I(T_I^{\uparrow n}) \\ T_I &= \bigcup_{n < \omega} T_I^{\uparrow n} \\ F_I^{\downarrow 0} &= HB(P) \\ F_I^{\downarrow n+1} &= \mathcal{F}_I(F_I^{\downarrow n}) \\ F_I &= \bigcap_{n < \omega} F_I^{\downarrow n}. \end{aligned}$$

A natural extension of the well-founded semantics to disjunctive programs might be defined as a fixpoint of an operator \mathcal{S}^D defined by $\mathcal{S}^D(S) = S \cup \langle T_S^D; F_S^D \rangle$ for any state S , where T_S^D and F_S^D are defined as follows:

Definition 3.5

$$\begin{aligned} T_S^{\uparrow 0} &= \emptyset \\ T_S^{\uparrow n+1} &= \mathcal{T}_S^D(T_S^{\uparrow n}) \\ T_S^D &= \bigcup_{n < \omega} T_S^{\uparrow n} \\ F_S^{\downarrow 0} &= CHB(P) \\ F_S^{\downarrow n+1} &= \mathcal{F}_S^D(F_S^{\downarrow n}) \\ F_S^D &= \bigcap_{n < \omega} F_S^{\downarrow n} \end{aligned}$$

□

Then, the well-founded state M_P^D for a disjunctive program P would be defined as M_δ where:

$$\begin{aligned} M_0 &= \langle \emptyset, \emptyset \rangle; \\ M_{\alpha+1} &= \mathcal{S}^D(M_\alpha), \text{ i.e. } M_{\alpha+1} = M_\alpha \cup \langle T_{M_\alpha}^D; F_{M_\alpha}^D \rangle \\ M_\alpha &= \bigcup_{\beta < \alpha} M_\beta, \text{ for a limit ordinal } \alpha. \end{aligned}$$

δ is the smallest countable ordinal such that M_δ is a fixpoint of the operator \mathcal{S} . In this case, the operator \mathcal{S} reduces to the operator \mathcal{I} when applied to general Horn programs. However, we will show through some examples that although \mathcal{S}^D and M_P^D extend the semantics in [RM88] by being able to capture the semantics of some unstratified disjunctive logic pro-

grams, the truth value of some formulas is undefined even though these formulas are false using the GCWA.

First we give some examples whose intended meaning is captured by M_P^D but is not captured by the stratification theory presented in [RM88]. In the examples we introduce two new notations. A set of disjuncts $\{D_1, \dots, D_n\}$ delimited by \parallel denotes the set of all disjuncts D that are subsumed by some D_i . A set of conjuncts $\{C_1, \dots, C_m\}$ delimited by $[]$ denotes the set of all conjuncts C that logically implies some conjunct C_i .

Example 3.3 For $HB(P) = \{a, b, c\}$:

$$\parallel a, b \vee c \parallel = \{a, a \vee b, a \vee c, b \vee c, a \vee b \vee c\}. [a, b \wedge c] = \{a, a \wedge b, a \wedge c, b \wedge c, a \wedge b \wedge c\}.$$

Example 3.4 Consider the following program

$$a \vee b \leftarrow \neg c, d$$

$$c \leftarrow \neg a, \neg b$$

Since this program is not stratified, its semantics is not captured in Minker and Rajasekar's stratification theory [RM88]. For this program we now compute the well-founded model state, as described above.

$$M_1 = \langle \emptyset, \emptyset \rangle \cup \langle \emptyset, [d] \rangle = \langle \emptyset, [d] \rangle \text{ since there is no rule with } d \text{ in its head.}$$

$$F_{M_1}^{10} = [a, b, c, d], \text{ the conjunctive Herbrand base}$$

$$F_{M_1}^{11} = \mathcal{F}_{M_1}^D(F_{M_1}^{10}) = [a, b, d], \text{ since the body of the rule with } c \text{ in its head is true with respect to the previous state}$$

$$F_{M_1}^D = F_{M_1}^{12} = F_{M_1}^{11}$$

$$M_2 = \langle \emptyset, [d] \rangle \cup \langle \emptyset, [a, b] \rangle = \langle \emptyset, [a, b, d] \rangle$$

$$M_3 = \langle \emptyset, [a, b, d] \rangle \cup \langle \parallel c \parallel, \emptyset \rangle = \langle \parallel c \parallel, [a, b, d] \rangle$$

$$M_P^D = M_3. \quad \square$$

Example 3.5 Consider the following disjunctive program without negation in the body of its rules.

$$a \vee b$$

$$c \vee p \vee b$$

$$p \leftarrow a$$

$$p \leftarrow b$$

We show that the direct extension of the well-founded semantics captures the GCWA for this program.

$$M_0 = \langle \emptyset, \emptyset \rangle$$

$$M_1 = \langle \parallel a \vee b, p \parallel, \emptyset \rangle$$

$$F_{M_1}^{10} = [a, b, c, p]$$

$$F_{M_1}^{11} = [c, p]$$

$$F_{M_1}^{12} = [c]$$

$$F_{M_1}^D = F_{M_1}^{13} = F_{M_1}^{12}$$

$$M_2 = \langle \parallel a \vee b, p \parallel, [c] \rangle$$

$$M_P^D = M_3$$

The results obtained by Minker and Rajasekar [MR90] capture the same semantics since the program is disjunctive and does not contain negated atoms in the body of any clause. \square

Example 3.6 Consider the following general Horn program:

$$p \leftarrow \neg a, \neg b, \neg c$$

$$a \leftarrow \neg b$$

$$b \leftarrow \neg a$$

$$c \leftarrow \neg d$$

$$d \leftarrow \neg c$$

The well-founded semantics of Van Gelder, Ross & Schilf [VRS88] assigns to the above program the partial model $\langle \emptyset, \emptyset \rangle$. Well-founded semantics [Prz89b] assumes that $\neg a \wedge \neg b$ is unknown. Hence it is not able to infer that p is false. The generalized disjunctive well-founded semantics of the program is $\langle \parallel a \vee b, c \vee d \parallel, [ab, cd, p] \rangle$. Hence, p is inferred to be false in the generalized disjunctive well-founded semantics. \square

The above example shows the extension of our semantics over well-founded semantics for general Horn programs. However, as the following example shows, it does not capture the GCWA for some disjunctive programs.

Example 3.7 Consider the following disjunctive logic program

$$p \leftarrow q, t$$

$$q \vee t$$

$$M_0 = \langle \emptyset, \emptyset \rangle$$

$F_{M_0}^{10} = [p, q, t]$, the conjunctive Herbrand base of the program.

$F_{M_0}^{11} = [p]$, since the body of the rules with q and t in the head are vacuously true, and the disjuncts that are present with q and t in the head of the rules are not true with respect to M_0

$F_{M_0}^{12} = \emptyset$, since the positive literals in the body of the rules with p in the head do not belong to $F_{M_0}^{11}$.

$$F_{M_0}^D = F_{M_0}^{13} = F_{M_1}^{12}$$

$$M_1 = \langle (q \vee t), \emptyset \rangle$$

$$M_P^D = M_2 = M_1$$

In this example even though p is false by the GCWA [Min82, MR90], the generalized well-founded semantics is not able to capture it. \square

To capture the natural semantics of programs of the type given in Example 3.7, while also handling the semantics of Example 3.5, we need a function similar to \mathcal{I}^E [BLM89] with two components. One of the components will be $\langle T_S^D; F_S^D \rangle$ and the other will be an extension of $\langle T_I^E; F_I^E \rangle$ to disjunctions and conjunctions. The definitions of T_I^E and F_I^E are based on a transformation of a general Horn program P to a positive disjunctive program $Dis(P, I)$. This transformation is obtained by removing false clauses in P with respect to I and moving negative literals in the body of each clause in P to its head. Then T_I^E and F_I^E are defined as follows:

$$T_I^E = \{A : A \text{ is an atom and } A \in T_{Dis(P, I)}^I \uparrow \omega \text{ and } A \notin I.\},$$

$$F_I^E = \{A : A \in GCWA(Dis(P, I)) \text{ and } A \notin I.\}.$$

$T_{Dis(P, I)}^I$ is the operator described in Definition 3.1. A precise description of the transformation $Dis(P, I)$ for general disjunctive programs and the extended definition of T_I^E and F_I^E are given in the next section. In this new definition, we use the extension of Minker's GCWA by Yahya and Henschen [YH85] to be able to infer the falsity of conjuncts. We call this extension the Generalized Closed World Assumption for Conjuncts (GCWAC). This semantics is a direct extension of the generalized well-founded semantics for general Horn programs [BLM89] to general disjunctive programs. We therefore call it the generalized disjunctive well-founded semantics for general disjunctive programs.

3.2 Generalized Disjunctive Well-Founded Semantics

We give two definitions for the GCWAC. The first, a syntactic definition based on proof theory. The second, a semantic definition based on model theory. The proof of equivalence can be found in [YH85].

Definition 3.6 (*Syntactic*)[YH85]

Let P be a disjunctive logic program and $C_1 \wedge \dots \wedge C_r$ a ground conjunct. Then, $C_1 \wedge \dots \wedge C_r$ can be inferred to be false from P iff $\forall K_1, \dots, K_r$ ($P \vdash C_1 \vee K_1, \dots, P \vdash C_r \vee K_r \Rightarrow P \vdash K_1 \vee \dots \vee K_r$). $GCWAC(P)$ is the set of conjuncts that can be assumed false from the program using the GCWAC. \square

Definition 3.7 (*Semantic*)[YH85]

Let P be a disjunctive logic program and $C_1 \wedge \dots \wedge C_r$ be a ground conjunct. Then, $C_1 \wedge \dots \wedge C_r$ can be inferred to be false from P iff $C_1 \wedge \dots \wedge C_r$ is false in every minimal model of P . \square

Although the GCWAC extends the information given by the GCWA the complexity of computing the GCWAC reduces to the computation of the GCWA as the following lemma shows.

Lemma 1 (*Relation between GCWA and GCWAC*)

Let P be a disjunctive logic program, C_* be a new ground atom not in $HB(P)$, and $C_1 \wedge \dots \wedge C_r$ be a ground conjunct, then

$C_1 \wedge \dots \wedge C_r \in GCWAC(P)$ iff $C_* \in GCWA(P \cup \{C_* \leftarrow C_1 \wedge \dots \wedge C_r\})$.

Proof:

$C_1 \wedge \dots \wedge C_r \in GCWAC(P)$

iff $C_1 \wedge \dots \wedge C_r$ is false in all minimal models of P

iff for every minimal model M of P there exists i , $1 \leq i \leq r$ such that C_i is false in M

iff C_* is false in every minimal model of P

iff C_* is false in every minimal model of $P \cup \{C_* \leftarrow C_1 \wedge \dots \wedge C_r\}$

iff $C_* \in GCWA(P \cup \{C_* \leftarrow C_1 \wedge \dots \wedge C_r\})$. \square

There are two properties of the GCWAC presented in [YH85] that we use later in the paper.

Theorem 2 (Maximal Consistency)[YH85]

Given a program P , $P \cup \neg GCWAC(P)$ is maximally consistent in the following sense: every conjunct C that can be assumed false without it being possible to derive a positive or empty clause not derivable from P belongs to $GCWAC(P)$. \square

Theorem 3 [YH85]

A Herbrand interpretation M is a minimal model of P if and only if M is a model of $P \cup \neg GCWAC(P)$. \square

Let S be a state and T_S^D and F_S^D be as defined in the previous subsection. Intuitively, T_S^D and F_S^D have the following meaning: T_S^D is the set of new positive ground clauses that can be derived from program P starting with S . F_S^D is the set of new atomic conjuncts that can be assumed false about P starting with S .

We now define the operators T_S^{ED} , F_S^{ED} and S^{ED} , where the superscript **ED** means we are Extending the well-founded semantics, and we are dealing with Disjunctive logic programs.

Definition 3.8 Let P be a general disjunctive program and S be a state. $DIS(P)$ is a disjunctive program obtained by transferring all negative literals in the body of clauses of P to its head. The disjunctive transformation of the program P with respect to S , $Dis(P, S)$, is a disjunctive program obtained from $DIS(P)$ by reducing the clauses in $DIS(P)$ as follows:

1. Remove atoms from the body of a clause if they are true in S .
2. Remove a clause if an atom or a disjunction in its head is true in S .
3. Remove the atoms in the head of a clause if they are false in S . \square

Definition 3.9 Let P be a program, S be a state and $Dis(P, S)$ be the disjunctive transformation of P with respect to S . T_S^{ED} and F_S^{ED} are defined as follows.

$$T_S^{ED} = \{C : C \in (T_{Dis(P, S)}^S \uparrow \omega) \text{ and } C \notin S\},$$

i.e. T_S^{ED} is the set of disjuncts that can be derived from $Dis(P, S) \cup S$.

$$F_S^{ED} = \{C : C \in GCWAC(Dis(P, S) \cup S) \text{ and } C \notin S\},$$

i.e. F_S^{ED} is the set of new ground conjuncts that can be assumed false about $Dis(P, S) \cup S$ by the GCWAC. \square

Definition 3.10 Let $\mathcal{S}^{\mathcal{ED}}$ be the operator that assigns to every state S of P a new state $\mathcal{S}^{\mathcal{ED}}(S)$ defined by :

$$\mathcal{S}^{\mathcal{ED}}(S) = S \cup \langle T_S^D; F_S^D \rangle \cup \langle T_S^{\mathcal{ED}}; F_S^{\mathcal{ED}} \rangle. \quad \square$$

Corollary 3.1 Given a state S , then $S \subseteq \mathcal{S}^{\mathcal{ED}}(S)$.

Proof: Directly from the definition of $\mathcal{S}^{\mathcal{ED}}$. \square

Definition 3.11 Let $M_0 = \langle \emptyset, \emptyset \rangle$;

$$M_{\alpha+1} = \mathcal{S}^{\mathcal{ED}}(M_\alpha),$$

$$\text{i.e. } M_{\alpha+1} = M_\alpha \cup \langle T_{M_\alpha}^D; F_{M_\alpha}^D \rangle \cup \langle T_{M_\alpha}^{\mathcal{ED}}; F_{M_\alpha}^{\mathcal{ED}} \rangle;$$

$$M_\alpha = \bigcup_{\beta < \alpha} M_\beta, \text{ for limit ordinal } \alpha. \quad \square$$

Since the sequence $\{M_\alpha\}$ of states is monotonically increasing there exists a smallest countable ordinal δ such that M_δ is a fixpoint, not necesarely a least fixpoint, of the operator \mathcal{S} . We call this fixpoint $M_P^{\mathcal{ED}}$.

We shall show in Corollary 3.3 that this fixpoint is consistent. That is, the union of the positive clauses and the negation of the conjuncts has a model. We now consider Example 3.7 in light of this extension to the well-founded semantics for disjunctive programs.

Example 3.8 The program we considered in Example 3.7 is:

$$p \leftarrow q, t$$

$$q \vee t$$

$$M_0 = \langle \emptyset, \emptyset \rangle$$

$$T_{M_0}^D = \parallel q \vee t \parallel$$

$$F_{M_0}^D = \emptyset$$

$\text{Dis}(P, M_0)$ has the clauses $\{p \leftarrow q, t; q \vee t\}$.

$$T_{M_0}^{\mathcal{ED}} = \parallel q \vee t \parallel$$

$F_{M_0}^{\mathcal{ED}} = [p, q \wedge t]$, as p and $q \wedge t$ both are false in all minimal models of $\text{Dis}(P, M_0)$.

$$M_1 = \langle \emptyset; \emptyset \rangle \cup \langle \parallel q \vee t \parallel, \emptyset \rangle \cup \langle \parallel q \vee t \parallel, [p] \rangle = \langle \parallel q \vee t \parallel, [p] \rangle$$

M_1 is the fixpoint, and it includes the semantics of the GCWA. \square

We now give another example which combines both the well-founded aspects and the GCWA.

Example 3.9 Consider the following program

$$\begin{aligned} p &\leftarrow t, q \\ q &\leftarrow \neg a \\ t &\leftarrow \neg b \\ a &\vee b \\ e &\leftarrow \neg f, p \\ f &\leftarrow \neg e \end{aligned}$$

First we compute the fixpoint using the direct extension of the well-founded semantics for disjunctive logic programs presented in Section 3.1.

$$M_0 = \langle \emptyset, \emptyset \rangle$$

We know $M_{\alpha+1} = M_\alpha \cup \langle T_{M_\alpha}^D; F_{M_\alpha}^D \rangle;$

$$T_{M_0}^D = \parallel a \vee b \parallel$$

$F_{M_0}^{10} = [a, b, p, q, t, e, f]$, the Conjunctive Herbrand base

$F_{M_0}^{11} = [p, e]$, since for all rules with p and e in their head the body is false with respect to $F_{M_0}^{10}$ and this is not the case for all rules with other atoms in their head.

$F_{M_0}^{12} = [e]$, since for all rules with e in the head the body is false with respect to $F_{M_0}^{11}$.

$F_{M_0}^{13} = \emptyset$, since for the rule with e in the head the body is no longer false with respect to $F_{M_0}^{12}$.

$$F_{M_0}^D = F_{M_0}^{14} = F_{M_0}^{13} = \emptyset$$

$$M_1 = \langle \emptyset, \emptyset \rangle \cup \langle \parallel a \vee b \parallel, \emptyset \rangle = \langle \parallel a \vee b \parallel, \emptyset \rangle$$

$$M_P^D = M_1$$

Hence the semantics assigns unknown to all atoms, except that it assigns true to the disjunct $a \vee b$. We now compute the fixpoint using the generalized disjunctive well-founded semantics.

$$M_0 = \langle \emptyset; \emptyset \rangle$$

$$T_{M_0}^D = \parallel a \vee b \parallel$$

$$F_{M_0}^D = \emptyset$$

$Dis(P, M_0)$ has the clauses $\{p \leftarrow t, q; q \vee a; t \vee b; a \vee b; e \vee f\}$.

$$T_{M_0}^{ED} = \parallel q \vee a, t \vee b, a \vee b, e \vee f \parallel.$$

$$F_{M_0}^{ED} = [p, q \wedge t, a \wedge b, q \wedge a, t \wedge b, e \wedge f].$$

$$M_1 = \langle \emptyset; \emptyset \rangle \cup \langle \parallel a \vee b \parallel, \emptyset \rangle \cup \langle \parallel q \vee a, t \vee b, a \vee b, e \vee f \parallel, [p, q \wedge t, a \wedge b, q \wedge a, t \wedge b, e \wedge f] \rangle \\ = \langle \parallel q \vee a, q \vee b, a \vee b, e \vee f \parallel, [p, q \wedge t, a \wedge b, q \wedge a, t \wedge b, e \wedge f] \rangle$$

$$T_{M_1}^D = \emptyset$$

$$F_{M_1}^D = [e]$$

$Dis(P, M_1)$ has the clauses $\{p \leftarrow t, q; q \vee a; t \vee b; a \vee b; e \vee f\}$.

$$T_{M_1}^{ED} = \emptyset.$$

$$F_{M_1}^{ED} = \emptyset.$$

$$M_2 = \langle \parallel q \vee a, t \vee b, a \vee b, e \vee f \parallel, [p, q \wedge t, a \wedge b, q \wedge a, t \wedge b, e \wedge f] \rangle \cup \langle \emptyset; [e] \rangle \cup \langle \emptyset, \emptyset \rangle \\ = \langle \parallel q \vee a, t \vee b, a \vee b, e \vee f \parallel, [p, q \wedge t, e, a \wedge b, q \wedge a, t \wedge b, e \wedge f] \rangle$$

$$\text{Similarly, } M_3 = \langle \parallel q \vee a, t \vee b, a \vee b, e \vee f \parallel, [p, q \wedge t, e, a \wedge b, q \wedge a, t \wedge b, e \wedge f] \rangle \cup \langle (f); \emptyset \rangle \\ \cup \langle \emptyset; \emptyset \rangle$$

$$= \langle \parallel q \vee a, q \vee b, a \vee b, f \parallel, [p, q \wedge t, e, a \wedge b, q \wedge a, t \wedge b, e \wedge f] \rangle$$

$M_4 = M_3$ is the fixpoint. □

3.3 Model Theoretic Semantics

In this section we study the relationship between the models of a program P and M_P^{ED} . In the same spirit as the priority relation defined by Przymusiński [Prz89b] for well-founded models we establish a priority relation between the minimal models of a general disjunctive program P . These priorities are defined in term of a stratification of the Herbrand base of P based on the syntactic structure of the program. The main result of this section is that the “perfect” models with respect to this relation are precisely the models of M_P^{ED} (see [Prz89b]).

Definition 3.12 Let P be a general disjunctive program and δ be the smallest (countable) ordinal such that M_δ is the fixpoint of \mathcal{S}^{ED} (see Definition 3.10). The dynamic stratification of the Herbrand base of P , $HB(P)$, is a partition of $HB(P)$ defined as follows:

$$S_\alpha = \{A \in HB(P) : A \in T_{M_\alpha}^D \cup F_{M_\alpha}^D \cup T_{M_\alpha}^{ED} \cup F_{M_\alpha}^{ED}, \text{ for } \alpha \leq \delta\}$$

$$S_{\delta+1} = HB(P) - \bigcup_{\alpha \leq \delta} S_\alpha.$$

We define \overline{S}_α to be the set $HB(P) - \bigcup_{\beta \leq \alpha} S_\beta$ and S_α^* to be the set $\bigcup_{\beta \leq \alpha} S_\beta$, for $\alpha \leq \delta + 1$. □

This stratification induces a priority relation \prec between the atoms of the Herbrand base of a program P . We say that an atom B has higher priority than an atom A ($A \prec B$) if A is in a lower stratum than B , (i.e. $(A \prec B)$ iff $A \in S_\alpha$, $B \in S_\beta$ and $\alpha < \beta$). Even though we follow a different approach than that given in [Prz89b] to define the priority relation among atoms in the $HB(P)$, this relation can be defined in terms of the dependency graph of the program P as in the well-founded semantics.

Next, we show how, using the priority relation, we can describe the generalized well-founded semantics by the perfect model semantics of Przymusiński [PP88, Prz88c]. First, we define the preference relation between models of P .

Definition 3.13 *Preference Relation [Prz88c]*

Let M and N be two distinct models. We say $N \ll M$ (N is preferable to M) if, $\forall A$ (a ground atom) in $N-M$, \exists a ground atom in B in $M-N$, such that $A \prec B$. We refer to the relation \ll as the generalized well-founded relation. \square

Example 3.10 Consider the program P in Example 3.4. The dynamic stratification of $HB(P) = \{a, b, c, d\}$ is: $S_1 = \{d\}$, $S_2 = \{a, b\}$ and $S_3 = \{c\}$. The minimal models of P are $\{a\}$, $\{b\}$ and $\{c\}$. By the priority relation, $\{c\} \ll \{a\}$ and $\{c\} \ll \{b\}$. Therefore, the perfect model of P is $\{c\}$. \square

It is not always the case that the preferred model of the relation \ll is unique. Consider the program in Example 3.8. The minimal models are $\{q\}$ and $\{t\}$. The dynamic stratification has only one stratum $S_1 = \{q, t, p\}$. Therefore, both models $\{q\}$ and $\{t\}$ are perfect models. We want to characterize the semantics of a general disjunctive program P through the minimal elements of the preference relation \ll . We define $Minset(P)$ to be the set of minimal elements with respect to the generalized well-founded relation \ll . An element in $Minset(P)$ is called a generalized well-founded model of P . A formula F is true (resp. false) with respect to the generalized well-founded models iff F is true (resp. false) in all the models in $Minset(P)$. Now, we have two structures to describe the semantics of a general disjunctive program P : the state M_P^{ED} and the generalized well-founded models in $Minset(P)$. The following definition allows us to transform sets of models into states and vice-versa.

Definition 3.14 $THREE(S)$

Let P be a program and S be a set of interpretations (models) for P . We define the state of P with respect to S , represented as $THREE(S)$, as a pair $\langle T', F' \rangle$, where

$$T' = \{C \in EHB(P) : \forall M \in S (M \models C)\}.$$

$$F' = \{C \in CHB(P) : \forall M \in S (M \models \neg C)\}.$$

$$THREE(MINSET(P, i)) = \langle T'_i, F'_i \rangle.$$

□

Example 3.11 From Example 3.8 we have:

$$THREE(\{\{q\}, \{t\}\}) = \langle \{q \vee t, q \vee t \vee p\}, \{p, p \wedge q, p \wedge q \wedge t\} \rangle$$

Next, we prove that $M_P^{ED} = THREE(Minset(P))$ by induction on the strata of P . We first extend the definition of $Minset$ to each stratum α in P .

Definition 3.15 $MINSET(P, \alpha)$

Let P be a program and $MM(P)$ be the set of minimal models of P . $MINSET(P, \alpha)$ is a set of minimal models of P defined by:

$$1. \forall M (M \in MM(P) \Rightarrow \exists N (N \in MINSET(P, \alpha) \wedge N \ll M).$$

$$2. \forall M, N (M, N \in MINSET(P, \alpha), M \neq N \Rightarrow M \not\ll N, N \not\ll M).$$

where the relation between the atoms $A, B \in HB(P)$ is defined by:

$A \prec B$ iff $A \in S_l$ and $B \in S_m$ and $l > m$ and $m \leq \alpha$; where $S_0 \dots S_{\delta+1}$ is the dynamic stratification of the program P .

$$Minset(P) = MINSET(P, \delta + 1), \text{ where } M_P^E = M_\delta.$$

□

A useful lemma that we need for the proofs is the following.

Lemma 2 $\alpha \leq \beta \Rightarrow MINSET(P, \beta) \subseteq MINSET(P, \alpha)$

Proof: Let $M \in MINSET(P, \beta)$ and assume $M \notin MINSET(P, \alpha)$. By definition of $MINSET(P, \alpha)$ there exists $N \in MINSET(P, \alpha)$ such that $N \ll_\alpha M$, where \ll_α is the relation defined by stratum α . Therefore, $\forall A \in N - M \exists B \in M - N (A \prec_\alpha B)$. Therefore, $\forall A \in S_l$ in $N - M$ for any stratum $l \exists B \in S_m$ in $M - N$ for some stratum $m \leq \alpha$, such that $l > m$. Therefore, $\forall A \in S_l$ in $N - M$ for any stratum $l \exists B \in S_m$ in $M - N$ for some stratum $m \leq \beta$, such that $l > m$, since $\alpha \leq \beta$. Then, $N \ll_\beta M$, where \ll_β is the relation

defined by stratum β . By definition of $MINSET(P, \beta)$, there exists $S \in MM(P)$ such that $S \ll_{\beta} N$ and $S \in MINSET(P, \beta)$. It is easy to show \ll_{β} is transitive. Hence, $S \ll_{\beta} M$, $S, M \in MINSET(P, \beta)$, $S \neq M$, contradicting part 2 of Definition 3.15. \square

Theorem 4 $M_P^{ED} = THREE(Minset(P))$

Proof: We prove this theorem by proving the following two lemmas:

1. Lemma 3: $M_{\alpha} \subseteq THREE(MINSET(P, \alpha))$.
2. Lemma 5: $THREE(MINSET(P, \alpha)) \subseteq M_{\alpha+1}$.

\square

Lemma 3 Given a general disjunctive program P and the sequences $\{M_{\alpha}\}_{\alpha=0}^{\infty}$.

$$M_{\alpha} \subseteq THREE(MINSET(P, \alpha)).$$

Proof: By induction, assume there is an ordinal β such that for all ordinals α , $\alpha < \beta$, $M_{\alpha} \subseteq THREE(MINSET(P, \alpha))$. Let $M_{\alpha} = \langle T_{\alpha}, F_{\alpha} \rangle$. Assume first that β is a successor ordinal $\alpha + 1$. We have two cases:

$$(1) C \in T_{\alpha+1} = T_{\alpha} \cup T_{M_{\alpha}} \cup T_{M_{\alpha}}^E.$$

(1.a) Assume $C \in T_{M_{\alpha}} = \cup_{i \leq \omega} T_{M_{\alpha}}^{\uparrow i}$. By induction on i , we prove that C is true in $THREE(MINSET(P, \alpha + 1))$. For $i = 0$, $T_{M_{\alpha}}^{\uparrow 0} = \emptyset$ and there is nothing to prove. Assume now as induction hypothesis that for every disjunct $D \in T_{M_{\alpha}}^{\uparrow i}$, for some i greater than 0, D is true in $THREE(MINSET(P, \alpha + 1))$. Let C be in $T_{M_{\alpha}}^{\uparrow i+1}$. Therefore, there exists a clause $C' \leftarrow B_1, \dots, B_m, \neg B_{m+1}, \dots, \neg B_k$ in P and a ground substitution θ such that for all j , $1 \leq j \leq m$, there is a disjunct C_j such that either $(B_j \vee C_j)\theta \in T_{\alpha}$ or $(B_j \vee C_j)\theta \in T_{M_{\alpha}}^{\uparrow i}$, $\{B_{m+1}, \dots, B_k\} \subseteq F_{\alpha}$ and the smallest factor of $(C' \vee C_1 \vee \dots \vee C_m)\theta$ subsumes C . By the general induction hypothesis if $(B_j \vee C_j)\theta$ is true in M_{α} then $(B_j \vee C_j)\theta$ is true in $THREE(MINSET(P, \alpha))$. Then, $(B_j \vee C_j)\theta$ is true in every model in $MINSET(P, \alpha)$. Therefore, by Lemma 2, $(B_j \vee C_j)\theta$ is true in every model in $MINSET(P, \alpha + 1)$. Similarly, by the general induction hypothesis and Lemma 2, $\{B_{m+1}, \dots, B_k\} \subseteq F_{\alpha+1}$. By the induction hypothesis in this part, if $(B_j \vee C_j)\theta \in T_{M_{\alpha}}^{\uparrow i}$ then $(B_j \vee C_j)\theta$ is true in

$MINSET(P, \alpha + 1)$. Hence, C is true in every model in $MINSET(P, \alpha + 1)$. Moreover, C is true in $THREE(MINSET(P, \alpha + 1))$.

(1.b) Let $C \in T_{M_\alpha}^E$. By definition of $T_{M_\alpha}^E$, C is a logical consequence of M_α . Therefore, by the general induction hypothesis $M_\alpha \subseteq THREE(MINSET(P, \alpha))$. Hence, C will be true in any minimal model in $MINSET(P, \alpha)$, since these models are also models of M_α . Then, C is true in every minimal model in $MINSET(P, \alpha + 1)$ by Lemma 2 making C true in $THREE(MINSET(P, \alpha + 1))$.

(1.c) It follows directly from Lemma 2 that C is true in $THREE(MINSET(P, \alpha + 1))$ when C is true in M_α .

(2) $C \in F_\alpha \cup F_{M_\alpha} \cup F_{M_\alpha}^E$.

(2.a) Assume $C \in F_{M_\alpha} = \bigcap_{i \leq \omega} F_{M_\alpha}^{li}$. By the definition of F_{M_α} if $C \in F_{M_\alpha}$ then there is a ground atom A occurring in C such that $A \in F_{M_\alpha}$. By the induction hypothesis we know that every model in $MINSET(P, \alpha)$ is a model of M_α . Then, by Lemma 2, every model in $MINSET(P, \alpha + 1)$ is a model of M_α . It is easy to prove that there exists a minimal model N of P that is a model of $M_{\alpha+1}$. We prove that for every minimal model M where A is true, $N \ll M$. Therefore, M cannot belong to $MINSET(P, \alpha + 1)$. Hence, C is false in $THREE(MINSET(P, \alpha + 1))$.

Let M be a minimal model of P such that A is true in M and M is a model of M_α . Since M belongs to $MINSET(P, \alpha + 1)$, M has to be a model of M_α (by the induction hypothesis and Lemma 2). Therefore, $N - M \subseteq \overline{S_\alpha}$ since N is a model of $M_{\alpha+1}$. We also know that $A \in M - N$ and A belongs to S_α . Therefore, by the order associated with the atoms in $HB(P)$ by $MINSET(P, \alpha + 1)$, for all atoms D in $N - M$, $D \prec A$. Then, $\forall D \in N - M \exists A \in M - N (D \prec A)$. Hence $N \ll M$.

(2.b) Let $C \in F_{M_\alpha}^E$. By Theorem 2, C is false in all minimal models of $Dis(P, M_\alpha)$ and by the induction hypothesis $M_\alpha \subseteq THREE(MINSET(P, \alpha))$, hence C is false in any minimal model in $MINSET(P, \alpha)$, as these models are models of M_α . Then, C is false in all minimal models in $MINSET(P, \alpha + 1)$ by Lemma 2, making C false in $THREE(MINSET(P, \alpha + 1))$.

(2.c) It follows directly from Lemma 2 that C is false in $THREE(MINSET(P, \alpha + 1))$ when C is false in M_α .

Consider the case when β is a limit ordinal. If a disjunct C is true in M_β then C is true in $\bigcup_{\alpha < \beta} T_{M_\alpha} \cup \bigcup_{\alpha < \beta} T_{M_\alpha}^E$. Then, C is true in $MINSET(P, \alpha)$, for some $\alpha < \beta$. (from (1.a) and (1.b)). Therefore, C is true in $THREE(MINSET(P, \beta))$, by Lemma 2. If a conjunct C is false in M_β then C is false in $\bigcup_{\alpha < \beta} F_{M_\alpha} \cup \bigcup_{\alpha < \beta} F_{M_\alpha}^E$. Then C is false in $MINSET(P, \alpha)$, for some $\alpha < \beta$. (from (2.a) and (2.b)). Therefore, C is false in $THREE(MINSET(P, \beta))$, by Lemma 2. \square

The following is a technical lemma used to prove Stattement 2 in Theorem 4.

Lemma 4 *If $M \in MINSET(P, \alpha)$ then M is a model for $T_{M_\alpha}^D \cup T_{M_\alpha}^{ED}$.*

Proof:

(1) M is a model of $T_{M_\alpha}^D = \bigcup_{i=0}^\omega T_{M_\alpha}^{Di}$. We prove by induction on i that M is a model for every element in $T_{M_\alpha}^D$. For the base case there is nothing to prove since $T_{M_\alpha}^{D0} = \emptyset$. As induction hypothesis assume that if $C \in T_{M_\alpha}^{Di}$ then $M \models C$, for some $i > 0$. Let $C \in T_{M_\alpha}^{D(i+1)}$. Then, there exists a ground instance of a clause in P , $C' \leftarrow B_1, \dots, B_m, \neg B_{m+1}, \dots, \neg B_n$, such that either $B_i \vee C_i \in T_{M_\alpha}^{Di}$ or $B_i \vee C_i$ is true in M_α for $1 \leq i \leq m$, $\neg B_i$ is true in M_α for $m+1 \leq i \leq n$ and C is subsumed by $C' \vee C_1 \vee \dots \vee C_m$. Since $M_\alpha \subseteq THREE(MINSET(P, \alpha))$ every minimal model of $THREE(MINSET(P, \alpha))$ is a model of M_α . Therefore, M is a model of M_α . Then,

1. if $B_i \vee C_i$ is true in M_α then $M \models B_i \vee C_i$.
2. if $\neg B_i$ is true in M_α then $M \models \neg B_i$.
3. if $B_i \vee C_i \in T_{M_\alpha}^{Di}$ then by the induction hypothesis $M \models B_i \vee C_i$.

Given that M is a model of P we have that $M \models C' \vee B_{m+1} \vee \dots \vee B_n \leftarrow B_1, \dots, B_m$. If $M \not\models C_i$ for all $1 \leq i \leq m$ then $M \models B_1 \wedge \dots \wedge B_m$ (by 1 & 2). Therefore, $M \models C'$ since $M \models \neg B_{m+1} \wedge \dots \wedge \neg B_n$ (by 3). Hence, $M \models C$ since C' subsumes C . If $M \models C_i$ for some $1 \leq i \leq m$ then $M \models C$ since C_i subsumes any clause that $C' \vee C_1 \vee \dots \vee C_m$ subsumes.

(2) M is a model of $T_{M_\alpha}^{ED}$. Similar to Case 1. \square

Lemma 5 *For any ordinal α , $THREE(MINSET(P, \alpha)) \subseteq M_{\alpha+1}$.*

Proof: $THREE(MINSET(P, \alpha)) \subseteq M_{\alpha+1}$ if and only if every (minimal) model M of $M_{\alpha+1}$ belongs to $MINSET(P, \alpha)$. Assume $T_\beta = \{A \in HB(P): A \text{ is true in } M_\beta\}$ and $F_\beta = \{A \in HB(P): A \text{ is false in } M_\beta\}$. Then, $T_{\alpha+1} \subseteq M$ and $F_{\alpha+1} \cap M = \emptyset$. Assume that there exists a model N such that $N \ll M$ and $N \in MINSET(P, \alpha)$. Since $M_\alpha \subseteq THREE(MINSET(P, \alpha))$ then $T_\alpha \subseteq N$ and $F_\alpha \cap N = \emptyset$. Therefore, $N - M \subseteq \overline{S_\alpha}$ and $M - N \subseteq \overline{S_\alpha}$. Moreover, $N - M \subseteq F_{\alpha+1} \cup \overline{S_{\alpha+1}}$ and $M - N \subseteq \overline{S_{\alpha+1}}$ by the previous lemma. Then, N cannot be preferable to M since the elements in $M - N$ are incomparable with the elements in $N - M$ contradicting our assumption. \square

Corollary 3.2 *Every model of M_P^{ED} is a minimal model of P .* \square

Corollary 3.3 *M_P^{ED} is a consistent state.*

Proof: We check that the computation at each step is consistent. i.e. we have to prove $(T_{M_\alpha}^D \cup T_{M_\alpha}^{ED} \cup \neg F_{M_\alpha}^D \cup \neg F_{M_\alpha}^{ED})$ is consistent, for all α . By Lemma 3 we have that:

$$M_\alpha \subseteq THREE(MINSET(P, \alpha)).$$

Therefore $M_{k+1} \subseteq THREE(MINSET(P, k+1))$. Then, the models in $THREE(MINSET(P, k+1))$ are models of $(T_{M_k}^D \cup T_{M_k}^{ED} \cup \neg F_{M_k}^D \cup \neg F_{M_k}^{ED})$. Hence $(T_{M_k}^D \cup T_{M_k}^{ED} \cup \neg F_{M_k}^D \cup \neg F_{M_k}^{ED})$ is consistent. \square

4 Procedural Semantics

In this section we present SLIS-resolution, a proof-procedure to answer queries on general disjunctive programs based on the generalized disjunctive well-founded semantics. This procedure is a modification of SLOC-resolution [MRL] and the procedure described by Lobo, Minker and Rajasekar in [LRM88] to compute answers in (positive) disjunctive programs using the Generalized Closed World Assumption and the Weak Generalized Closed World Assumption, respectively. All these proof-procedures are based on SLI-resolution [MZ82, MR88], an extension of SL-resolution [KK71] that allows arbitrary literal selection. The main structures used in SLIS-resolution are t-clauses.

Definition 4.1 *A general t-clause Γ , is an ordered triple $\langle T, A, C \rangle$ where*

- **T** is a labeled tree. The root is labeled with the distinguished symbol ϵ , and the other nodes are labeled with literals; and
- **A** and **C** are marking relations on the nodes such that every non-terminal node belongs to the **A** marking relation. \square

Literals with **A** marking are called A-literals and literals with **C** marking C-literals. Unmarked literals are called B-literals. We will use “*” and “+” to mark A-literals and C-literals respectively. A general disjunctive clause is represented by a general t-clause where every negative literal in the body is marked as a C-literal, the remaining literals are B-literals and the distinguished symbol ϵ occurs in the t-clause.

Example 4.1 ($\epsilon^* P Q \neg R S^+$) is a general t-clause representation of the program clause $P \vee Q \leftarrow R \wedge \neg S$ where P, Q and R are B-literals and S is a C-literal. \square

More complex t-clauses will be created by the derivation steps described below. The idea is to keep the ancestry information of each literal in the derived clause in the t-clause structure. A general t-clause can be viewed as a pre-order representation of a resolution tree. We next give a formal definition for SLIS-resolution. First, we define two sets of literals which are used in the definitions.

$\gamma_L = \{M: \text{where } M \text{ is a B-literal, and } M \text{ is one node off the path from the root } \epsilon^* \text{ of } L\}$

$\delta_L = \{N: \text{where } N \text{ is an A-literal, and } N \text{ is on the path between the root } \epsilon^* \text{ and } L\}$

A t-clause is said to satisfy the admissibility condition (AC) if for every occurrence of every B-literal L in it the following conditions hold:

- (i) No two literals from γ_L and L have atoms which unify.
- (ii) No two literals from δ_L and L have atoms which unify.

If we look at the t-clauses as resolution trees, we say that a t-clause τ satisfies the minimality condition (MC) if there is no A-literal in τ which is a leaf or terminal node. γ_L and δ_L are used while performing factoring and ancestry resolution respectively. Definitions for factoring and ancestry resolution are given below¹.

Definition 4.2 Consider a general t-clause C_0 . Then C_n is a tranfac – derivation (truncation, ancestry and factoring) of C_0 when there is a sequence of general t-clauses C_0, C_1, \dots, C_n

¹Ancestry and factoring are similar to that in SL-resolution

such that for all i , $0 \leq i \leq n$, C_{i+1} is obtained from C_i by either t -factoring, t -ancestry, or t -truncation.

C_{i+1} is obtained from C_i by t -factoring iff

- (1) C_i is $(\alpha_1 L \alpha_2 M \alpha_3)$ or C_i is $(\alpha_1 M \alpha_2 L \alpha_3)$;
- (2) L and M have the same sign and unify with mgu θ ;
- (3) L is in γ_M (i.e., L is in an higher level of the tree);
- (4) C_{i+1} is $(\alpha_1 L \alpha_2 \alpha_3)\theta$ or C_{i+1} is $(\alpha_1 \alpha_2 L \alpha_3)\theta$

C_{i+1} is obtained from C_i by t -ancestry iff

- (1) C_i is $(\alpha_1 (L^* \alpha_2 (\alpha_3 M \alpha_4) \alpha_5) \alpha_6)$;
- (2) L and M are complementary and unify with mgu θ ;
- (3) L is in δ_M ;
- (4) C_{i+1} is $(\alpha_1 (L^* \alpha_2 (\alpha_3 \alpha_4) \alpha_5) \alpha_6)\theta$;

C_{i+1} is obtained from C_i by t -truncation iff

- either C_i is $(\alpha (L^*) \beta)$ and C_{i+1} is $(\alpha \beta)$.
or C_i is (ε^*) and C_{i+1} is \square .

The underlying idea behind the generalized disjunctive well-founded semantics stems from the fact that for any model M of a general disjunctive program P that assigns true to a set of atoms $\{B_1, \dots, B_m\}$ that occurs in a program clause of the form:

$$A_1 \vee \dots \vee A_k \leftarrow B_1, \dots, B_m, \neg B_{m+1}, \dots, \neg B_n,$$

M also assigns true to the disjunction $A_1 \vee \dots \vee A_k \vee B_{m+1} \vee \dots \vee B_n$. This allows us to use any variation of a rule, where we move a (not necessarily proper) subset of the negated atoms in the body to the head in the resolution process.

Example 4.2 Consider the program:

$$\begin{aligned} p &\leftarrow a \\ p &\leftarrow b \\ a &\leftarrow \neg b \\ b &\leftarrow \neg a \end{aligned}$$

we can deduce p from the variant $a \vee b$ of the rule $a \leftarrow \neg b$.

A formal definition for variant clauses is the following:

Definition 4.3 Given a general t -clause $\Gamma = (\epsilon^* A_1 \dots A_n \neg B_1 \dots \neg B_m C_1^+ \dots C_p^+)$ where $n + m + p \geq 1$ and $n \geq 0, m \geq 0, p \geq 0$. A **mark variant** of Γ is a t -clause of the form $\Gamma' = (\epsilon^* A'_1 \dots A'_n C'_1 \dots C'_q \neg B'_1 \dots \neg B'_m C_{q+1}'^+ \dots C_p'^+)$ where for $1 \leq i \leq n$ A'_i is a variable variant of A_i , for $1 \leq j \leq m$, B'_j is a variable variant of B_j and $\{C'_1, \dots, C'_q\} \cup \{C_{q+1}', \dots, C_p'\}$ forms a partition (modulo variable renaming) of $\{C_1, \dots, C_p\}$. $C'_i, 1 \leq i \leq q$ are called the **mark variant atoms** of Γ . \square

In addition, two rules of negation are used in the process of derivation: the GCWAC and a minimization rule that assumes an atom false if all the rules that define the atom, (i.e. rules with occurrences of the atom in the head) can be assumed false. Roughly speaking, the former rule allows us to prove d in the program $P_1 = \{d \leftarrow \neg c; c \leftarrow a, b; c \vee e \leftarrow f; a \vee b; e; f\}$. The latter allows us to prove p from the program $P_2 = \{p \leftarrow \neg q; q \leftarrow m; q \leftarrow s\}$.

In the process of proving d and p we have to prove that c is false in P_1 and that q is false in P_2 . To prove c is false we use the first rule of negation. Therefore, we have to prove for all positive clauses of the form $a_1 \vee \dots \vee a_n \vee c$ that we can derive from P_1 we can also derive $a_1 \vee \dots \vee a_n$. In this case, the only clauses of this form are $a \vee b \vee c$ and $c \vee e$. So, if $a \vee b$ and e are true then we can assume $\neg c$. To prove q false we use the second rule. This means that we have to prove that there is no rule with q in its head such that the body of the rule is true. In this case, we have to prove that m and s are false in order to assume q false.

We use two different structures in the proof procedure to implement the two rules of negation: $+$ trees whose leaves will contain the positive clauses needed to be proved true in order to assume the query (in our case this will be the negation of the query) true and SLISNF trees whose leaves will contain the clauses needed to be proved false in order to assume the (negation) of the query true. In the following definitions we introduce the concept of the rank of an SLIS-refutation, a $+$ -tree and an SLISNF tree. This definition is irrelevant for the procedure but it will be used in the proof of soundness.

Definition 4.4 Given a general program P , an **SLIS-derivation** from a set of t -clauses $T = \{T_1, \dots, T_s\}$ is a (possibly infinite) sequence of general t -clauses (C_1, C_2, \dots) such that C_1 is a transac-derivation of a clause in T and for each $C_i = (\epsilon^* \alpha_1 L \beta_1)$ either
(1) L is a B -literal and $C = (\epsilon^* \alpha_2 M \beta_2)$ such that either C is clause in P or a mark variant of a clause in P with M as one of the mark variant atoms and

1. L and M are complementary and unify with mgu θ
2. C'_{i+1} is $(\epsilon^* \alpha_1 (L^* \alpha_2 \beta_2) \beta_1) \theta$.
3. C_{i+1} is a tranfac-derivation of C'_{i+1} .
4. C_{i+1} satisfies the admissibility and minimality conditions.

or

(2) L is a ground C-literal A^+ , there exists a successful (positive tree) $+tree$, or a finitely failed SLISNF tree, for $(\epsilon^* \neg A)$ and $C_{i+1} = (\epsilon^* \alpha_1 \beta_1)$ (see Definitions 4.5 & 4.7 for the definitions of $+trees$ and SLISNF trees respectively).

An **SLIS-refutation** is a finite SLIS-derivation (C_1, C_2, \dots, C_n) such that $C_n = \square$. If C_{i+1} is obtained via Case 1, C_{i+1} has the rank of C_i . If C_{i+1} is obtained via Case 2 and $(\epsilon^* \neg A)$ has a $+tree$ or a failed SLISNF tree of rank κ . The rank of C_{i+1} is the maximum between $\kappa + 1$ and the rank of C_i . C_1 has rank 1. The SLIS-refutation (C_1, C_2, \dots, C_n) is a refutation of rank κ iff κ is the rank of C_n . \square

Before presenting the definition of $+trees$ we need to define positive selection functions. A selection function is positive if the function selects only negative B-literals. This function is called positive because negative B-literals in a t-clause represent positive literals in the body of a disjunctive clause (see Example 4.1). The interest of positive selection functions comes from the definition of the GCWAC where we want to derive all possible positive clauses. The clause formed using the B-literals and the C-literals at any step of an SLIS-derivation is a logical consequence of the program and the set of t-clauses T . Using a positive selection function in a derivation will result in clauses with only positive B-literals and C-literals. Proving that the clause formed with the B-literals is a logical consequence of the program allows us to assume the initial goal to be false by the GCWAC. SLIS is based on SLI-resolution which is complete and sound for an arbitrary selection function. Hence, it is complete and sound for a positive selection function. The completeness and soundness of SLIS-resolution is based on this and the GCWA.

Definition 4.5 Given a program P , a positive selection function R and general t-clause G , a $+tree$, T_G^+ is defined inductively for G as follows:

(1) The root of T_G^+ is G .
(2) For a node $D = (\epsilon^* \alpha_1 L \beta_1)$, assume L is selected by R . Then, there is a child H of D for each t -clause $C = (\epsilon^* \alpha_2 M \beta_2)$ such that either C is a clause in P or a mark variant of a clause in P with M as one of the mark variant atoms and:

1. L and M are complementary and unify with mgu θ
2. H' is $(\epsilon^* \alpha_1 (L^* \alpha_2 \beta_2) \beta_1) \theta$.
3. H is a tranfac-derivation of H' .
4. H satisfies the admissibility and minimality conditions. □

There are three classes of leaves in a $+$ tree (resp. SLISNF) tree.

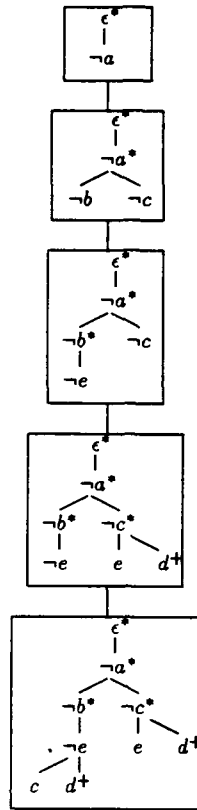
1. Empty leaves which contain the empty clause \square .
2. Dead leaves which contain negative B-literals that do not match any clause in P .
3. Goal leaves which contain only positive B-literals and C-literals in the frontier. We denote the set of B-literals of a goal leaf G by $B(G)$, and the set of C-literals by $C(G)$.

Definition 4.6 If T_G^+ contains an empty leaf then T_G^+ is a failure $+$ tree. Otherwise, if all the leaves are dead leaves or goal leaves and for each goal leaf G there is an SLIS-refutation for $\{(\epsilon^* \neg A_1), \dots, (\epsilon^* \neg A_n), (\epsilon^* \neg B_1), \dots, (\epsilon^* \neg B_m)\}$ from P , where $B(G) = \{A_1, \dots, A_n\}$ and $\{B_1, \dots, B_m\}$ is the subset of ground atoms in $C(G)$, then T_G^+ is a **successful** $+$ tree. In any other case T_G^+ is an undefined tree. A dead leaf has rank 1. A goal leaf G has rank κ iff the SLIS-refutation for $\{(\epsilon^* \neg A_1), \dots, (\epsilon^* \neg A_n), (\epsilon^* \neg B_1), \dots, (\epsilon^* \neg B_m)\}$ has rank κ . T_G^+ is of rank κ iff κ is the least upper bound of the ranks associated to the leaves of T_G^+ . □

Example 4.3 Let P be the following disjunctive program:

$$\begin{array}{l} a \leftarrow b, c \\ b \leftarrow e \\ e \vee c \leftarrow \neg d \end{array}$$

The following is a successful $+$ tree for $\neg a$:

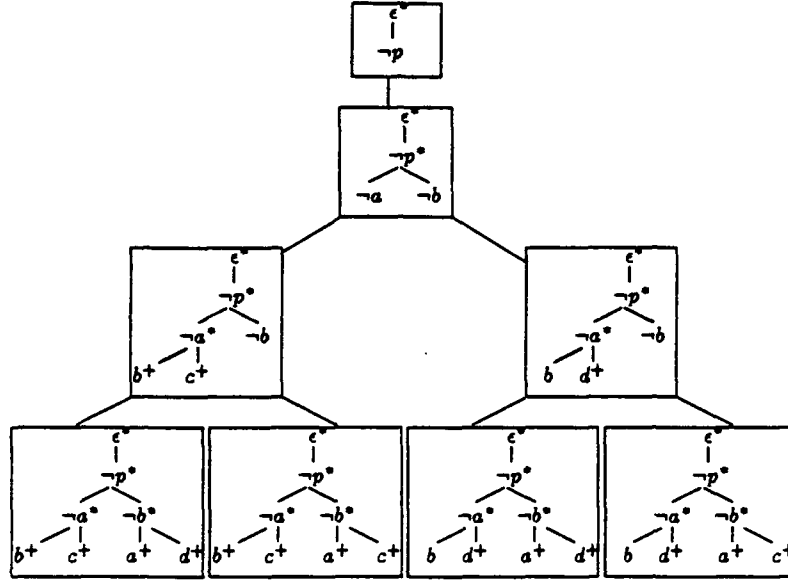


We can obtain an SLIS-refutation for $\{(\epsilon^* \neg c), (\epsilon^* \neg d), (\epsilon^* \neg e)\}$ from the third rule of the program.

Example 4.4 Let P be the following disjunctive program:

$p \leftarrow a, b$
 $a \leftarrow \neg b, \neg c$
 $b \leftarrow \neg a, \neg d$
 $c \vee d$

The following is a successful +tree for $\neg p$:



There are four more leaf nodes in the tree that are obtained from mark variants that are not presented in the figure. But since in $+trees$ C -literals and positive B -literals are handled in the same way these nodes will have the same $SLIS$ -refutation as the leaf nodes in the figure. Finally, it is easy to see that there are $SLIS$ -refutations for $\{(\epsilon^* \neg b), (\epsilon^* \neg c), (\epsilon^* \neg a), (\epsilon^* \neg d)\}$ and $\{(\epsilon^* \neg b), (\epsilon^* \neg d), (\epsilon^* \neg a)\}$ using the third clause in the program.

Definition 4.7 Given a program P , a ground atom A and a positive selection function R . A $SLISNF$ tree, T_G^S for $G = (\epsilon^* \neg A)$ is defined as follows:

- (1) The root of T_G^S is G .
- (2) For a node $D = (\epsilon^* \alpha_1 L \beta_1)$, assume L is selected by R . Then, there is a child H of G for each t -clause $C = (\epsilon^* M D_1 \dots D_k \neg B_1 \dots \neg B_m B_{m+1}^+ \dots B_n^+)$ in P such that:

1. L and M are complementary and unify with mgu θ
2. H' is $(\epsilon^* \alpha_1 (L^* \alpha_2 \beta_2) \beta_1) \theta$.
3. H is a tranfac-derivation of H' .
4. H satisfies the admissibility and minimality conditions.

If all leaves are either dead leaves or goal leaves and for all goal leaves, one of the following conditions is satisfied:

- Then T_G^S is a failed SLISNF tree. If T_G^S contains an empty leaf then T_G^S is a successful SLISNF tree. In any other case T_G^S is an undefined tree. All dead leaves are rank 1 leaves. T_G^S is of rank κ iff κ is the least upper bound of the ranks associated with the leaves of T_G^S . \square

$$\begin{array}{l} a \leftarrow b, c \\ b \leftarrow e \\ e \vee c \leftarrow d \\ f \leftarrow \neg d \end{array}$$

```

      c
      |
      a
     / \
    b   c
    |   | \
    e   e  d
   / \
  c   d

```

$$m(X) \leftarrow \neg q(Y), \neg p(Y)$$

$$p(a) \vee q(a) \quad r(b)$$

If we build a +tree starting with $(\epsilon^* \neg m(a))$, the only goal leaf that we obtain is:

$(\epsilon^* (\neg m(a)^* q(Y)^+ p(Y)^+))$. Using the second rule in the program we can find an SLIS-refutation for $\{(\epsilon^* \neg q(Y)), (\epsilon^* \neg p(Y))\}$. Therefore, If we allow non-ground atoms in the subset of $C(G)$ used for the SLIS-refutation on the leaves in +trees, we are able to find a successful +tree for $(\epsilon^* \neg m(a))$ even though $m(a)$ is true in M_P^{ED} .

Example 4.7 Let P be:

$$\begin{aligned} m(X) &\leftarrow q(Y), \neg p(Y) \\ q(Y) \\ p(a) \\ r(b) \end{aligned}$$

If we build a +tree starting with $(\epsilon^* \neg m(a))$, the only goal leaf that we obtain (before the tranfac-derivation) is: $(\epsilon^* (\neg m(a)^* (q(Y)^* p(Y)^+))$. If we allow non-ground atoms in the subset of $C(G)$ used for the SLIS-refutation on the leaves in SLISNF trees, we are able to find a failed SLISNF tree for $(\epsilon^* \neg m(a))$ even though $m(a)$ is true in M_P^{ED} .

We will find SLIS-derivations that are stopped because the ground condition does not hold for any subset of $C(G)$. We call these derivations *floundering* derivations. SLIS-refutations allow us to prove when either a positive (not necessarily ground) clause or a negated ground atom is true in M_P^{ED} . More general queries can be obtained combining these two types of queries. We also have to assume that the queries have a non-floundering SLIS-refutation. We call a query with a non-floundering derivation a non-floundering query.

Theorem 5 (Soundness) Let P be a general disjunctive program, $Q = A_1 \vee \dots \vee A_n$ be a positive clause and $L = \neg A$ be a negative literal. Let Q and L be non-floundering queries.

1. If there is an SLIS-refutation for $\{(\epsilon^* \neg A_1) \dots (\epsilon^* \neg A_n)\}$ from P then $M_P^{ED} \models \exists Q$.
2. If there is an SLIS-refutation for $\{(\epsilon^* A^+)\}$ from P then $M_P^{ED} \models L$.

Proof: We prove by induction on the rank of a refutation, that :

1. If there is an SLIS-refutation for $\{(\epsilon^* \neg A_1) \dots (\epsilon^* \neg A_n)\}$ from P , and the SLIS refutation with minimum rank has rank κ , then $M_\kappa \models \exists Q$.
2. If there is an SLIS-refutation for $\{(\epsilon^* A^+)\}$ from P and the SLIS refutation with minimum rank has rank $\kappa + 1$, then $M_\kappa \models L$. By definition of SLIS refutation, an SLIS-refutation of rank $\kappa + 1$ for $\{(\epsilon^* A^+)\}$ from P , means that there is either:
 - (a) a successful +tree of rank κ for $\{(\epsilon^* \neg A)\}$ from P .
 - or
 - (b) a failed SLISNF tree of rank κ for $\{(\epsilon^* \neg A)\}$ from P .

$\kappa = 1$

(1) Since an SLIS refutation of rank 1 is an SLI refutation, this is true by the soundness of SLI refutation [MZ82].

(2a) A successful +tree of rank 1 for $\{(\epsilon^* \neg A)\}$, means all leaves of the +tree are either goal leaves or dead leaves and for all goal leaves G , where $B(G) = \{A_1, \dots, A_n\}$ and $\{B_1, \dots, B_m\}$ is the subset of ground atoms in $C(G)$; and there is an SLIS-refutation of rank 1, for $\{(\epsilon^* \neg A_1) \dots (\epsilon^* \neg A_n)\}$ from P . Due to the similarity of the structure between +trees of rank 1 and SLINF trees [MR88] this means that for all $A \vee K$ that have a SLIS refutation of rank 1 there is a SLIS-refutation of rank 1 for K . By the equivalence of the syntactic and semantic definitions of the GCWA [Min82], this means A is false in all minimal models of P , and hence $A \in F_{M_0}^D$; i.e. $M_1 \models L$.

(2b) A failed SLISNF tree of rank 1 for $G = \{(\epsilon^* \neg A)\}$ means all leaves of the SLISNF tree are dead leaves. The proof for this case is similar to the inductive case.

Induction Hypotheses: Statements 1 and 2 are true for all $\kappa < \mu$.

$\kappa = \mu$

(1) There is an SLIS-refutation for $\{(\epsilon^* \neg A_1) \dots (\epsilon^* \neg A_n)\}$ from P , and the SLIS refutation with minimum rank has rank μ . This means there is a finite SLIS-derivation (C_1, C_2, \dots, C_n) such that $C_n = \square$ and the rank of C_n is μ . Let i be the largest integer such that the rank of C_i is less than μ . Then C_{i+1} is obtained from C_i by Case 2 of Definition 4.4, where $C_i = (\epsilon^* \alpha_1 \ L \ \beta_1)$, L is a ground C-literal B^+ , μ is a successor ordinal and there exists a successful +tree of rank $\mu - 1$, or a failed SLISNF tree of rank $\mu - 1$, for $(\epsilon^* \neg B)$ and $C_{i+1} = (\epsilon^* \alpha_1 \ \beta_1)$. By part 2 of the induction hypotheses this means $M_{\mu-1} \models \neg B$ which

implies $M_\mu \models \neg B$. Using the information $M_\mu \models \neg B$ (as an oracle), we can assume the rank of C_{i+1} to be equal to the rank of C_i , i.e. less than μ . Therefore, the rank of $C_n = \square$ will be less than μ since the rank of C_n is same as the rank of C_{i+1} , and by the induction hypotheses, $M_\mu \models \exists Q$.

(2a) There is a successful +tree of rank μ for $\{(\epsilon^* \neg A)\}$ from P . That means all the leaves of the +tree are either dead or goal leaves and for all goal leaves G , where $B(G) = \{A_1, \dots, A_n\}$ and $\{B_1, \dots, B_r\}$ is the subset of ground atoms in $C(G)$ there is an SLIS-refutation of rank μ or less than μ , for $\{(\epsilon^* \neg A_1), \dots, (\epsilon^* \neg A_n), (\epsilon^* \neg B_1), \dots, (\epsilon^* \neg B_r)\}$ from P and the lub of these ranks is μ . By the induction hypotheses and part(1) of the proof when $\kappa = \mu$, $M_\mu \models A_1 \vee \dots \vee A_n \vee B_1 \vee \dots \vee B_r$, for all cases when $M_1 \models A \vee A_1 \vee \dots \vee A_n \vee B_1 \vee \dots \vee B_r$, which means A is false in M_μ by Lemma 6. i.e $M_\mu \models L$ where, $L = \neg A$.

Lemma 6 *Let A be a ground atom and m be an ordinal larger or equal than 1. Then $\exists A_1 \vee \dots \vee A_r$, a ground clause, such that $(M_1 \models A \vee A_1 \vee \dots \vee A_r)$ but $M_m \not\models A_1 \vee \dots \vee A_r$ iff A is true in some minimal model of M_m .*

Proof (similar to the proof in [She88])

(\Rightarrow)

$M_m \not\models A_1 \vee \dots \vee A_r$ means, $A_1 \vee \dots \vee A_r$ is false in some minimal model M of M_m . Since M also belongs to M_1 , and $A \vee A_1 \vee \dots \vee A_r$ is true in all minimal models in M_1 , $A \vee A_1 \vee \dots \vee A_r$ is also true in M . But as $A_1 \vee \dots \vee A_r$ is false in M , hence A must be true in M .

(\Leftarrow)

Let A be true in a minimal model M of M_m , and let $\{A_1, A_2, \dots\}$ be the set of all ground atoms false in M . Then, as M is a minimal model of M_1 , $M_1 \cup \{\neg A_1, \neg A_2, \dots\} \cup \{\neg A\}$ is inconsistent, (otherwise M will not be minimal in M_1). By compactness some finite subset is inconsistent, i.e., for some finite r ,

$M_1 \cup \{\neg A_1, \dots, \neg A_r\} \cup \{\neg A\}$ is inconsistent, which means $M_1 \models A \vee A_1 \vee \dots \vee A_r$. But $M_m \not\models A_1 \vee \dots \vee A_r$ because $A_1 \vee \dots \vee A_r$ is false in M . \square

(2b) There is a failed SLISNF tree of rank μ for $\{(\epsilon^* \neg A)\}$ from P . That means all leaves of the SLISNF tree $T_{\{(\epsilon^* \neg A)\}}^S$ are goal leaves or dead leaves and for all goal leaves, one of the following conditions is satisfied.

1. There is an SLIS-refutation of rank less than μ for $\{(\epsilon^* \neg D_1) \dots (\epsilon^* \neg D_k)\}$ from P , where $\{D_1 \dots D_k\} \subseteq B(G)$ and have the same parent.
2. There is an SLIS-refutation of rank less than μ for $\{(\epsilon^* \neg B_{m+1}) \dots (\epsilon^* \neg B_n)\}$ from P , where $\{B_{m+1}^+ \dots B_n^+\} \subseteq C(G)$, each B_i is a ground atom and have the same parent.
3. There is a successful +tree of rank less than μ for $(\epsilon^* \neg B_1 \dots \neg B_m)$ where $\neg B_1^* \dots \neg B_m^*$ are marked nodes in the goal leaf and have the same parent.

The lub of the ranks of the goal leaves is μ . We have to prove now that $M_\mu \models L$, where $L = \neg A$. Consider the set $U = \{ C : \exists \text{ a failed SLISNF tree of rank } \mu \text{ for } \{(\epsilon^* \neg C)\} \}$. By the induction hypotheses and the definition of a failed SLISNF tree of rank μ ,

$U =$

$\{ C : T_{\{(\epsilon^* \neg C)\}}^S \text{ has only goal leaves or dead leaves and for all goal leaves there is an ordinal } \nu < \mu \text{ such that one of the following conditions is satisfied:}$

1. $D_1 \vee \dots \vee D_k$ is true in M_ν , where $\{D_1 \dots D_k\} \subseteq B(G)$ and have the same parent.
2. $B_{m+1} \vee \dots \vee B_n$ is true in M_ν , where $\{B_{m+1}^+ \dots B_n^+\} \subseteq C(G)$, each B_i is a ground atom and have the same parent.
3. $B_1 \wedge \dots \wedge B_m$ is false in M_ν , where $\neg B_1^* \dots \neg B_m^*$ are marked nodes in the goal leaf and have the same parent.

$\}$

Let V be any arbitrary element of U . We claim that for all nodes N in $T_{\{(\epsilon^* \neg V)\}}^S$, there is an ordinal $\nu < \mu$ such that one of the following condition is satisfied:

1. $B_{m+1} \vee \dots \vee B_n$ is true in M_ν , where $\{B_{m+1}^+ \dots B_n^+\} \subseteq C(V)$, each B_i is a ground atom and have the same parent.
2. $B_1 \wedge \dots \wedge B_m$ is false in M_ν , where $\neg B_1 \dots \neg B_m$ are marked nodes in N where each B_i may or may not be marked, and have the same parent.

3. $B_1 \wedge \dots \wedge B_m$ is in $F_{M_\nu}^D$, where $\neg B_1 \dots \neg B_m$ are marked nodes in N , where each B_i may or may not be marked, and have the same parent.
4. $D_1 \vee \dots \vee D_k$ is true in M_ν , where $\{D_1 \dots D_k\} \subseteq B(V)$ and have the same parent.

We prove the above claim by contradiction. Suppose there is a node N_1 in $T_{\{\epsilon^* \neg V\}}^S$, where none of the above conditions is satisfied. This would mean for each $B_i, 1 \leq i \leq m$ there will be a goal leaf of $T_{\{\epsilon^* \neg V\}}^S$, that is a descendant of N_1 , where the conditions required to be in U will be violated and hence V will not be in U making our assumption false. Since V is any arbitrary element of U , the above claim is true for all elements of U . Since A is in U , the above claim is true with respect to A also and by definition of $F_{M_\nu}^D$, $A \in F_{M_\nu}^D$. Hence $M_\mu \models L$, where $L = \neg A$. \square

We need the definition of ground SLISNF trees for completeness proof of SLIS-resolution

Definition 4.8 *Given a program P and positive selection function R . A ground SLISNF tree, T_G^g for G is defined as follows:*

- (1) *The root of T_G^g is G .*
- (2) *For a node $D = (\epsilon^* \alpha_1 L \beta_1)$, assume L is selected by R . Then, there is a child H of G for each t -clause, $C = (\epsilon^* M D_1 \dots D_k \neg B_1 \dots \neg B_m B_{m+1}^+ \dots B_n^+)$, ground instance of a t -clause in P such that:*

1. *L and M are complementary and unify.*
2. *H' is $(\epsilon^* \alpha_1 (L^* \alpha_2 \beta_2) \beta_1)$.*
3. *H is a tranfac-derivation of H' .*
4. *H satisfies the admissibility and minimality conditions.*

If all leaves are either dead leaves or goal leaves and for all goal leaves, one of the following conditions is satisfied:

1. *There is an SLIS-refutation for $\{(\epsilon^* \neg D_1) \dots (\epsilon^* \neg D_k)\}$ from P , where $\{D_1 \dots D_k\} \subseteq B(G)$ and have the same parent.*
2. *There is an SLIS-refutation for $\{(\epsilon^* \neg B_{m+1}) \dots (\epsilon^* \neg B_n)\}$ from P , where $\{B_{m+1}^+ \dots B_n^+\} \subseteq C(G)$, each B_i is a ground atom and have the same parent.*

3. $(\epsilon^* \neg B_1 \dots \neg B_m)$ has a successful \vdash -tree, where $\neg B_1^* \dots \neg B_m^*$ are marked nodes in the goal leaf and have the same parent.

Then T_G^S is a ground failed SLISNF tree. If T_G^S contains an empty leaf then T_G^S success SLINSF tree. In any other case T_G^S is an undefined tree. \square

Theorem 6 (Completeness) Let P be a general disjunctive program, $Q = A_1 \vee \dots \vee A_n$ be a positive clause and $L = \neg A$ be a negative literal. Let Q and L be non-floundering queries.

1. If $M_P^{ED} \models \exists Q$ then there is an SLIS-refutation for $\{(\epsilon^* \neg A_1) \dots (\epsilon^* \neg A_n)\}$ from P .
2. If $M_P^{ED} \models L$ then there is an SLIS-refutation for $\{(\epsilon^* A^+)\}$ from P .

Proof: According to Definition 3.11, $M_P^E = \bigcup_{\alpha < \delta} M_\alpha$ for some (countable) ordinal δ . We prove the theorem by induction on α . When $\alpha = 0$, $M_\alpha = \langle \emptyset, \emptyset \rangle$ and the theorem follows directly. Assume for an ordinal α and non-floundering queries $Q = A_1 \vee \dots \vee A_n$ and $L = \neg A$, if $\beta < \alpha$ and $M_\beta \vdash \exists Q$ and $M_\beta \vdash L$ then there exists SLIS-refutations for $\{(\epsilon^* \neg A_1) \dots (\epsilon^* \neg A_n)\}$ and $\{(\epsilon^* A^+)\}$ from P . Suppose now, there are non-floundering queries $Q = A_1 \vee \dots \vee A_n$ and $L = \neg A$ such that $M_\alpha \vdash \exists Q$ and $M_\beta \vdash L$ but $M_\beta \not\vdash \exists Q$ and $M_\beta \not\vdash L$ for any $\beta < \alpha$.

Case 1: $Q = A_1 \vee \dots \vee A_n$. Then $M_\alpha \vdash \exists Q$. Therefore, there is a finite set of ground substitutions $\{\theta_1, \dots, \theta_l\}$ such that $M_\alpha \vdash Q\theta_1 \vee \dots \vee Q\theta_l$. If α is a limit ordinal then $\bigcup_{\beta < \alpha} M_\beta \vdash Q\theta_1 \vee \dots \vee Q\theta_l$. Hence, there exists $\beta < \alpha$ such that $M_\beta \vdash Q\theta_1 \vee \dots \vee Q\theta_l$. Then $M_\beta \vdash \exists Q$ contradicting our assumption that $M_\beta \not\vdash \exists Q$ for any $\beta < \alpha$. If α is a successor ordinal $\beta + 1$ then $M_\beta \cup \langle T_{M_\beta}, F_{M_\beta} \rangle \cup \langle T_{M_\beta}^E, F_{M_\beta}^E \rangle \vdash Q\theta_1 \vee \dots \vee Q\theta_l$. Therefore $Q\theta_1 \vee \dots \vee Q\theta_l \in T_{M_\beta} \cup T_{M_\beta}^E$.

(a) $Q\theta_1 \vee \dots \vee Q\theta_l \in T_{M_\beta} = \bigcup_{n < \omega} T_{M_\beta}^{in}$. By induction on n we prove there is an SLIS-refutation for $Q\theta_1 \vee \dots \vee Q\theta_l$. The base case follows immediately since $T_{M_\beta}^{i0} = \emptyset$. Assume that for every $C \in T_{M_\beta}^{in}$ there is an SLIS-refutation and suppose $Q\theta_1 \vee \dots \vee Q\theta_l \in T_{M_\beta}^{in+1}$. Therefore, there exists a rule $A_1 \vee \dots \vee A_s \leftarrow B_1, \dots, B_k, \neg L_1, \dots, \neg L_m$ in the program P where the A s, B s and L s are atoms and a ground substitution δ such that for all i , $1 \leq i \leq k$, such that $B_i\delta \vee C_i$ either belongs to T_β or belongs to $T_{M_\beta}^{in}$, for all i , $1 \leq i \leq m$, $L_i\delta \in F_\beta$ and $A_1 \vee \dots \vee A_s \vee C_1 \vee \dots \vee C_k$ subsumes $Q\theta_1 \vee \dots \vee Q\theta_l$. Assume without lose of generality

that $k = m = 1$ and $(A_1 \vee \dots \vee A_s \vee C_1)\delta = Q\theta_1 \vee \dots \vee Q\theta_l$. Then, we start the refutation with P and the set $T = \{(\epsilon^* \neg A_1), \dots, (\epsilon^* \neg A_s), (\epsilon^* \neg C_1)\}$. Let $(\epsilon^* \neg A_1 \theta)$ be the initial t-clause in the derivation. In the first derivation step we use $(\epsilon^* A_1 \dots A_s \neg B_1 L_1^+)$ as the mark variant clause and obtain the tranfac-derivation of $(\epsilon^* (\neg A_1^* \theta \dots A_s \neg B_1 L_1^+))\sigma$, with σ the most general unifier between $A_1 \theta$ and A_1 . By the induction hypotheses we know that there are SLIS-refutations for $T' = \{(\epsilon^* \neg B_1)\delta, (\epsilon^* \neg C_1)\}$ and $(\epsilon^* L_1^+)\delta$. Hence, there are SLIS-refutations for $\{(\epsilon^* \neg B_1)\sigma, (\epsilon^* \neg C_1)\}$ and $(\epsilon^* L_1^+)\sigma$ since σ is more general than δ and $Q\theta_1 \vee \dots \vee Q\theta_l$ is a non-floundering query since Q is non-floundering. We can use these derivations (or more specific instantiations of these derivations) to construct a refutation for $(\epsilon^* (\neg A_1^* \theta \dots A_s \neg B_1 L_1^+))\sigma$.

(b) $Q\theta_1 \vee \dots \vee Q\theta_l \in T_{M_\beta}^E$. Since $T_{M_\beta}^E = T_{Dis(P, M_\beta)}^I \uparrow \omega$, this part can be derived from the completeness of SLI-resolution with respect to the operator $T_{Dis(P, M_\beta)}^I$. For that, assume we have the disjunctive program $Dis(P, M_\beta)$. By Theorem 1 and completeness of SLI-resolution [MZ82] there exists an SLI-refutation for Q since $Q\theta_1 \vee \dots \vee Q\theta_l$ belongs to $T_{Dis(P, M_\beta)}^I \uparrow \omega$. From the SLI-refutation of $Q\theta_1 \vee \dots \vee Q\theta_l$ using $Dis(P, M_\beta)$ we can build an SLIS-refutation using P . First, notice that for each clause $C = H \vee H_1 \vee \dots \vee H_m \leftarrow B_1, \dots, B_k$ in $Dis(P, M_\beta)$ where H is a positive clause and the H_i and B_i are atoms, there is a clause

$$C' = H \leftarrow B_1, \dots, B_k, B_{k+1}, \dots, B_l, \neg H_1, \dots, \neg H_m, \neg H_{m+1}, \dots, \neg H_q$$

in P from where C was obtained. We follow the same derivation steps that are used in the SLI-refutation for the SLIS-refutation of $Q\theta_1 \vee \dots \vee Q\theta_l$ but, instead of using clauses C from $Dis(P, M_\beta)$ we will use the mark variant:

$$H \vee H_1 \vee \dots \vee H_m \leftarrow B_1, \dots, B_k, B_{k+1}, \dots, B_l, \neg H_{m+1}, \dots, \neg H_q$$

of C' . From the definition of $Dis(P, M_\beta)$ we obtain at the end of the SLIS-derivation a general t-clause τ with ground C-literals C^+ (since Q is non-floundering) such that the C s are false in M_β and negated B-literals $\neg B$ such that the B s are true in M_β . Therefore, by the induction hypothesis of the general proof we have SLIS-refutations for each C-literal and B-literal in τ . Hence, we have a refutation for τ and therefore, a refutation for $Q\theta_1 \vee \dots \vee Q\theta_l$.

Case 2: L is a negative literal $\neg A$. Then $M_\alpha \vdash \neg A$. If α is a limit ordinal then $\bigcup_{\beta < \alpha} M_\beta \vdash \neg A$.

Then, there exists $\beta < \alpha$ such that $M_\beta \vdash \neg A$ contradicting our assumption. If α is a successor ordinal $\beta + 1$ then $M_\beta \cup \langle T_{M_\beta}, F_{M_\beta} \rangle \cup \langle T_{M_\beta}^E, F_{M_\beta}^E \rangle \vdash \neg A$. Therefore $A \in F_{M_\beta} \cup F_{M_\beta}^E$.

(a) $A \in F_{M_\beta} = \bigcap_{n < \omega} F_{M_\beta}^{I_n}$. We prove there is a ground SLISNF tree for $(\epsilon^* \neg A)$ and we lift this proof to the non-ground case. For each ground instance $(\epsilon^* A E_1 \cdots E_k \neg B_1 \cdots \neg B_m B_{m+1}^+ \cdots B_n^+)$ of a program clause, we obtain a child H of the form $(\epsilon^* (\neg A^* E_1 \cdots E_k \neg B_1 \cdots \neg B_m B_{m+1}^+ \cdots B_n^+))$. If there is no such rule, the node $(\epsilon^* \neg A)$ is a dead leaf. Therefore, $(\epsilon^* \neg A)$ is directly a failed SLISNF tree for A . Otherwise, since $A \in F_{M_\beta}$ one of the following cases holds for H :

1. $(B_{m+1} \vee \cdots \vee B_n) \in T_S$.
2. $(B_1 \wedge \cdots \wedge B_m) \in F_S$.
3. $(B_1 \wedge \cdots \wedge B_m) \in F$.
4. $(E_1 \vee \cdots \vee E_k) \in T_S$.

If Case 1 holds then every node N below H contains $\{B_{m+1}^+, \dots, B_n^+\} \subseteq C(N)$, since any selection function used to build a SLISNF tree is a positive selection function. In particular, when N is a leaf node $\{B_{m+1}^+, \dots, B_n^+\} \subseteq C(N)$. Therefore, by the induction hypothesis, there is an SLIS-refutation for $\{(\epsilon^* \neg B_{m+1}), \dots, (\epsilon^* \neg B_n)\}$ from the program P for all the leaves under H .

Similarly, if Case 4 holds then every node N below H contains $\{E_1, \dots, E_k\} \subseteq B(N)$, since any selection function used to build an SLISNF tree is a positive selection function. In particular, when N is a leaf node $\{E_1, \dots, E_k\} \subseteq B(N)$. Therefore, by the induction hypothesis, there is an SLIS-refutation for $\{(\epsilon^* \neg E_1), \dots, (\epsilon^* \neg E_k)\}$ from the program P for all the leaves under H .

If Case 2 holds, we have two subcases, either

- (a) There is a failed SLISNF-tree T for $(\epsilon^* B_i)$ for some i , $1 \leq i \leq m$; or
- (b) There is a successful $+$ -tree for $(\epsilon^* \neg B_1 \cdots \neg B_m)$.

If (a) is true, we select B_i and attach T as the SLISNF subtree under H . If (b) is true then every leaf below H contains a t-clause where B_1, \dots, B_m are marked literals of the same parent and there is a successful $+$ -tree for $(\epsilon^* \neg B_1 \cdots \neg B_m)$.

If Case 3 holds then there is an atom B_i for some i , $1 \leq i \leq m$ such that $B_i \in F_{M_\beta}$. We select B_i to construct a SLISNF tree under H using the same procedure described for A .

By construction, all the leaves in this SLISNF tree are either dead leaves or leaves in which one of the three conditions for failed SLISNF tree holds. Therefore, this tree is a failed SLISNF tree for $(\epsilon^* \neg A)$. By the Lifting Lemma below (see Lemma 7) every ground instance N' of leaf N in the SLISNF tree T_G^S constructed using the selection function for the ground failed SLISNF tree T_G^g , is a leaf in T_G^g . Therefore, T_G^g cannot be a failed tree for $(\epsilon^* \neg A)$, contradicting our assumption about T_G^g .

(b) $A \in F_{M_\beta}^E$. Similar to Part (b) in Case 1 but using the completeness result for the Support-for-Negation rule with respect to the GCWA and Lemma 1. These two cases complete the proof of the theorem. \square

Lemma 7 (Lifting Lemma) *Given a ground atom A , let $G = (\epsilon^* \neg A)$. For every node N in an SLISN tree T_N^S constructed using any positive selection function R , there is a node N' , a ground instance of N , in the ground SLISNF tree T_G^g constructed using R . Moreover, all the ground instances of N are nodes in T_G^g .*

Proof: Let t_1, \dots, t_n be the t-clauses used in the path from G to N . Let $\theta_1, \dots, \theta_n$ be the mgus used in each step. Assume θ to be the composition of these substitutions, i.e. $\theta = \theta_1 \dots \theta_n$. Let δ be a ground substitution such that $\delta = \theta\gamma$ for some substitution γ . Then, by definition of ground SLISNF trees there is a path from G to $N\delta$ using ground instances of t-clauses $t_1\delta, \dots, t_n\delta$. Since the selection of γ is independent of the construction, N' can be any instance of N . \square

To prove the general case (i.e. the non ground case), by contradiction, assume that G does not have a failed SLISNF-tree and let T_G^g be the ground failed tree for G . $(\epsilon^* \neg A)$ does not have a successful tree, otherwise SLIS would not be sound (contradicting Theorem 5). Therefore, every SLISNF tree for $(\epsilon^* \neg A)$ is undefined. This means that for each SLISNF tree there is a leaf G such that none of the following conditions holds:

1. There is an SLIS-refutation for $\{(\epsilon^* \neg D_1) \dots (\epsilon^* \neg D_k)\}$ from P , where $\{D_1 \dots D_k\} \subseteq B(G)$ and have the same parent.

2. There is an SLIS-refutation for $\{(\epsilon^* \neg B_{m+1}) \dots (\epsilon^* \neg B_n)\}$ from P , where $\{B_{m+1}^+ \dots B_n^+\} \subseteq C(G)$, each B_i is a ground atom and have the same parent.
3. $(\epsilon^* \neg B_1 \dots \neg B_m)$ has a successful $+$ -tree, where $\neg B_1^* \dots \neg B_m^*$ are marked nodes in the goal leaf and have the same parent. \square

During an SLIS-derivation, we have to choose among different mark variants to perform a derivation step. For example, given the two t -clauses $(\epsilon^* \neg p)$ and $(\epsilon^* q_1 q_2 \neg q_3 p^+ q_4^+)$, we can use either $(\epsilon^* q_1 q_2 \neg q_3 p^+ q_4^+)$ or $(\epsilon^* q_1 q_2 \neg q_3 p^+ q_4)$ to obtain either $(\epsilon^*(\neg p^* q_1 q_2 \neg q_3 q_4^+))$ or $(\epsilon^*(\neg p^* q_1 q_2 \neg q_3 q_4))$. There is no indication which of the two will lead us to the refutation. A simple rule, based on syntactic characteristics of the program, allows us to reduce the number of mark variants to consider in a refutation. Before presenting the rule we need the following definition.

Definition 4.9 *Given a general disjunctive program P ,*

1. *A relation Q refers positively (resp. negatively) to a relation R if there is a clause in P with Q occurring in the head and R occurring either in the head or a positive (resp. negated) atom in the body.*
2. *The dependency graph of a program P is the directed graph representing the relation refers.* \square

With the relation **refers** we can obtain mutual dependencies between definitions of predicates: two relations R and P are mutually dependent iff they belong to the same strongly-connected component in the dependency graph. Mark variants are needed only when the dependency between the mark variant atoms and the positive atoms in the clause is negative.

Definition 4.10 *(Restricted Mark Variants) Given a disjunctive program P and a general t -clause $\Gamma = (\epsilon^* A_1 \dots A_n \neg B_1 \dots \neg B_m C_1^+ \dots C_p^+)$ where $n + m + p \geq 1$ and $n \geq 0$, $m \geq 0$, $p \geq 0$. A mark variant of Γ is a t -clause of the form*

$\Gamma' = (\epsilon^ A'_1 \dots A'_n C'_1 \dots C'_q \neg B'_1 \dots \neg B'_m C'_{q+1}^+ \dots C'_p^+)$ where for $1 \leq i \leq n$ A'_i is a variable variant of A_i , for $1 \leq j \leq m$, B'_j is a variable variant of B_j , $\{C'_1, \dots, C'_q\} \cup \{C'_{q+1}, \dots, C'_p\}$*

forms a partition (modulo variable renaming) of $\{C_1, \dots, C_p\}$ and the relation symbols occurring in $\{C'_1, \dots, C'_q\}$ belong to the same strongly-connected component of some relation symbol occurring in $\{A_1, \dots, A_n\}$. C'_i , $1 \leq i \leq q$ are called the mark variant atoms of Γ . \square

The following example shows how the rule is used.

Example 4.8 Let P be a program:

- (1) $p \leftarrow q, \neg a$
- (2) $q \leftarrow s$
- (3) $a \leftarrow b, c$
- (4) $b \leftarrow e$
- (5) $e \vee c \leftarrow \neg d$
- (6) s .

The SLIS-refutation for $(\epsilon^* \neg p)$ is the following:

$(\epsilon^* \neg p)$ resolving with rule 1 we obtain

$(\epsilon^*(\neg p^* \neg q a^+))$ resolving with rule 2 we obtain

$(\epsilon^*(\neg p^* (\neg q^* \neg s) a^+))$ resolving with rule 6 after tranfac-derivation we obtain

$(\epsilon^*(\neg p^* a^+))$ using the +tree in Example 4.3 and after tranfac-derivation we obtain

\square .

In this example, we do not have to consider the variant $p \vee a \leftarrow q$ since p and a belong to different components in the dependency graph of P . \square

Other important simplifications can be obtained in the use of mark variants to construct +trees. Since while constructing +trees we use a positive selection function and in the leaves of the trees we treat C-literals as positive B-literals we have to consider only the mark variant whose unique mark atom is the one to which we apply resolution. This constraint clearly reduces the branches in many +trees. We refer to these variants as selected mark variants. In Example 4.4 if we use selected mark variants the +tree will only have two leaves: the first and the third leaves, counting left to right.

5 Comparison with other approaches

In this section we compare the generalized disjunctive well-founded semantics to Ross's strong well-founded semantics [Ros] and to Przymusinski's stationary model semantics [Prz]. In Section 5.1 we describe stationary models, in Section 5.2 we describe the strong well-founded semantics and we compare these with the generalized disjunctive well-founded semantics in Section 5.3.

5.1 Stationary Semantics

Przymusinski [Prz] presents 3-valued stationary models of disjunctive logic programs, as a generalization of 3-valued stable models for normal programs. His stationary semantics of disjunctive logic programs is based on the set of all 3-valued stationary models of the program.

Definition 5.1 ([Prz]) *Let M be any Herbrand interpretation of the program P and consider the reduced program P^* obtained from P by using the Gelfond-Lifschitz reduction, namely, by:*

- (i) *Removing from P all clauses which contain some negative premise $\neg B$ such that B is true in M ;*
- (ii) *Removing all negative premises from the remaining clauses.*

M is a STATIONARY model of P if M is a minimal model of P^ (Note P^* is a positive disjunctive program).* □

A 3-valued interpretation of P is a pair $I = \langle T; F \rangle$, where T and F are disjoint subsets of $HB(P)$. The atoms in T are called true in I , those in F are false in I and those in $U = HB(P) - (T \cup F)$ are undefined. It is assumed that t is always in T , u in U and f in F . If I is a 3-valued interpretation, then the truth value $val_I(C)$ of a ground atom C is defined as 1 (resp. 1/2 or 0) if C is in T (resp. U or F). For a negative literal $\neg C$, $val_I(\neg C) = 1 - val_I(C)$.

Definition 5.2 ([Prz]) *A 3-valued interpretation $I = \langle T; F \rangle$ is a model of a disjunctive program P if for every clause $A_1 \vee \dots \vee A_m \leftarrow L_1, \dots, L_n$ in P , $\max\{val_I(A_i) : i \leq m\} \geq \min\{val_I(L_i) : i \leq n\}$.* □

Definition 5.3 ([Prz]) *If I and I' are two 3-valued interpretations of P then I is said to be less than or equal to I' if $val_I(A) \leq val_{I'}(A)$, for every atom A . \square*

Minimal 3-valued models with respect to the above ordering are defined in the usual way.

Definition 5.4 ([Prz]) **3-valued Stationary Models**

Let M be any 3-valued interpretation of a disjunctive logic program P . Replace every negative premise $L \leftarrow C$ in every clause in P by “ t ” if L is true in M , by “ u ” if L is undefined in M and by “ f ” if L is false in M . As a result we obtain a positive disjunctive program P^ . We say that M is a 3-valued stationary model of P if and only if M is a minimal 3-valued model of P^* . \square*

Definition 5.5 ([Prz]) *The STATIONARY SEMANTICS of a disjunctive program consists of all closed formulae of the language that are true in all 3-valued stationary models of P . \square*

5.2 Strong Well-Founded Semantics (SWFS)

The strong well-founded semantics (SWFS) of disjunctive logic programs [Ros] is based on global trees and strong derivations.

Definition 5.6 ([Ros]) *Let Q and Q' be sets of extended literals (disjunctions or negated disjunctions). Q' is said to be strongly derived from Q (written $Q \Leftarrow Q'$) if Q contains a disjunction D and there is some instantiated rule R given by $H \leftarrow R_1 \wedge \dots \wedge R_j \wedge \neg S_1 \wedge \dots \wedge \neg S_k$, such that either*

$$(S_1) \ H \subseteq D \text{ and } Q' = (Q - \{D\}) \cup \{R_1 \vee D, \dots, R_j \vee D, \neg S_1, \dots, \neg S_k\}$$

or

$$(S_2) \ H \not\subseteq D, H \cap D \neq \emptyset \text{ (say } c = H - D), \text{ and } Q' = (Q - \{D\}) \cup \{R_1, \dots, R_j, \neg S_1, \dots, \neg S_k, \neg C\}$$

\square

Intuitively $Q \Leftarrow Q'$ means “if all of Q' is true then all of Q is also true.” If the last element of a finite strong derivation sequence for disjunct D is empty or has only negative extended literals then the last element is called the basis for D .

Definition 5.7 ([Ros]) *The strong global tree Γ_D^s for a given disjunction D , has the root as D and for any internal node D' of Γ_D^s has children as disjunctions which are of the form,*

\overline{Q} , where Q ranges over all bases for D . If $Q = \{\neg L_1, \dots, \neg L_n\}$ is an arbitrary set of ground negative literals then \overline{Q} is the disjunction $L_1 \vee \dots \vee L_n$.

Truth assignments of ground disjunctions in Γ_D^* is defined as follows:

1. if every child of D' is true, then D' is false.
2. If some child of D' is false, then D' is true.
3. all other nodes are undefined. □

Using the above definition a disjunction D is inferred to be true (false or undefined) by the strong well-founded semantics if D is assigned true (false or undefined) in Γ_D^* .

5.3 Comparison

Example 5.1 Consider the program P consisting of the clauses:

$a \leftarrow \neg b$

$a \leftarrow \neg c$

$b \vee c$

It has three minimal models $\{a, b\}$, $\{a, c\}$ and $\{b, c\}$. Among these three $\{a, b\}$ is a stationary model as $\{a, b\}$ is a minimal model of the corresponding $P^* = \{a, b \vee c\}$. Similarly $\{a, c\}$ is a stationary model. But $\{b, c\}$ is not a stationary model as the corresponding $P^* = \{b \vee c\}$. The 3-valued interpretations $M_1 = \langle \{a, b\}, \{c\} \rangle$ and $M_2 = \langle \{a, c\}, \{b\} \rangle$ are the 3-valued stationary models of the program. From this it is clear that a is true in all 3-valued stationary models of the program, P .

Considering the strong well-founded semantics of Ross [Ros], in Γ_a^* with respect to the above program a has two children b and c , as a has two bases $\neg b$ and $\neg c$. The children of b is c and c is b and this continues infinitely, as the basis of b is $\neg c$ and the basis of c is $\neg b$. Hence a is inferred to be undefined by the strong well-founded semantics of Ross [Ros]. Intuitively this is because it uses a "rule at a time" transformation of sets of extended literals.

In this example, the $\text{Minset}(P)$ of generalized well-founded models is equal to the set of stationary models. a and $b \vee c$ which are true in GDWFS are also true in all the 3-valued stationary models of the program. Also these are the only clauses that are true in all the 3-valued stationary models of P . Similarly, $b \wedge c$ is the only conjunct that is false in all the 3-valued stationary models of P , and are assigned false in GDWFS of P . Hence,

$THREE(\text{Minset}(P)) = \langle \parallel a, b \vee c \parallel, [bc] \rangle$. □

Example 5.2 Consider a program P consisting of the clauses:

$a \leftarrow \neg b$
 $b \leftarrow \neg a$
 $p \leftarrow a$
 $p \leftarrow b$
 $c \vee d \leftarrow p$

P has six 3-valued stationary models $M_1 = \langle \emptyset, \{c\} \rangle$, $M_2 = \langle \emptyset, \{d\} \rangle$, $M_3 = \langle \{p, a, c\}, \{b, d\} \rangle$, $M_4 = \langle \{p, a, d\}, \{b, c\} \rangle$, $M_5 = \langle \{p, b, c\}, \{a, d\} \rangle$, and $M_6 = \langle \{p, b, d\}, \{a, c\} \rangle$. M_1 is a 3-valued stationary model of P because the corresponding P^* is $\{a \leftarrow u, b \leftarrow u, p \leftarrow a, p \leftarrow b, c \vee d \leftarrow p\}$. The minimal model of P^* has a, b and p as unknown. To be a model of $c \vee d \leftarrow p$, $c \vee d$ has to be unknown. Hence the minimal model will have one of c and d as unknown and the other as false. Hence $\langle \emptyset, \{c\} \rangle$ is a minimal model of P^* , making it a 3-valued stationary model of P . The case for M_2 is similar.

M_3 is a 3-valued stationary model of P because the corresponding P^* is $\{a, p \leftarrow a, p \leftarrow b, c \vee d \leftarrow p\}$. Hence, a and p are true in the minimal models of P^* , one of c and d is true and the other is false in the minimal models of P^* , and b is false in the minimal models of P^* . Hence $\langle \{p, a, c\}, \{b, d\} \rangle$ is a minimal model of P^* making it a 3-valued stationary model of P . The case for M_4, M_5 and M_6 are similar.

Considering the strong well-founded semantics of Ross [Ros], in Γ_p^s with respect to the above program, p has two children a and b . This is because p has two bases $\neg a$ and $\neg b$ through the following two strong derivation sequences.

$\{p\} \Leftarrow \{p \vee a\} \Leftarrow \{p \vee a \vee b\} \Leftarrow \{\neg b\}$ and
 $\{p\} \Leftarrow \{p \vee b\} \Leftarrow \{p \vee a \vee b\} \Leftarrow \{\neg a\}$. The child of a is b and of b is a and this continues infinitely, as the basis of a is $\neg b$ and the basis of b is $\neg a$. Hence p is undefined in Γ_p^s . For similar reasons $c \vee d$ is undefined in $\Gamma_{c \vee d}^s$. But if the first two rules in the program are replaced by the clause $a \vee b$ then the derivation sequence of p will be $\{p\} \Leftarrow \{p \vee a\} \Leftarrow \{p \vee a \vee b\} \Leftarrow \emptyset$ and p will assigned true.

The $\text{Minset}(P)$ of generalized well-founded models is the set $\{\{a, p, c\}, \{b, p, c\}, \{a, p, d\}, \{b, p, d\}\}$. In this case then, $THREE(\text{Minset}(P)) = \langle \parallel a \vee b, p, c \vee d \parallel, [ab, cd] \rangle$.

Hence, for the program P , neither $c \vee d$ nor p is true in all the 3-valued stationary models, and the strong well-founded semantics of Ross, while they are true in the generalized disjunctive well-founded semantics. \square

6 Future work

Przymusiński [Prz88b, Prz88a, Prz89a] studies the relation between non-monotonic reasoning and different semantics of logic programs. In particular he studies the relationship between well-founded semantics and 3-valued formalisms of non-monotonic reasoning. He proves that the well-founded model of a program P is the intersection of all models of prioritized circumscription, $\text{CIRC3}(P; S_0 > \dots > S_\delta)$, where $\{S_\alpha\}_{\alpha \leq \delta}$ is the dynamic stratification of P , based on the well-founded semantics. Presently we are studying the relationship between a different 3-valued non-monotonic formalism and generalized disjunctive well-founded semantics of disjunctive logic programs. Our 3-valued formalism of non-monotonic reasoning is based directly on the set of models (2-valued) of the theory. In particular we are studying the relationship between M_P^{ED} and $\text{THREE}(S)$, where S is the set of all models of prioritized circumscription, $\text{CIRC2}(P; S_0 > \dots > S_\delta)$, where $\{S_\alpha\}_{\alpha \leq \delta}$ is the dynamic stratification based on the generalized disjunctive well-founded semantics.

Acknowledgements

We wish to express our appreciation to the National Science Foundation for their support of our work under grant number IRI-86-09170 and the Army Research Office under grant number DAAG-29-85-K-0-177.

References

- [BLM89] C. Baral, J. Lobo, and J. Minker. Generalized well-founded semantics for logic programs. Technical Report CS-TR-2330, Dept of Computer Science, University of Maryland, College Park Md 20742, 1989. To be presented in CADE 90.

- [GL88] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In R.A. Kowalski and K.A. Bowen, editors, *Proc. 5th International Conference and Symposium on Logic Programming*, pages 1070–1080, Seattle, Washington, August 15–19, 1988.
- [KK71] R. A. Kowalski and D. Kuehner. Linear Resolution with Selection Function. *Artificial Intelligence*, 2:227–260, 1971.
- [Llo84] J.W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1984.
- [LRM88] J. Lobo, A. Rajasekar, and J. Minker. Weak Completion Theory for Non-Horn Programs. In R.A. Kowalski and K.A. Bowen, editors, *Proc. 5th International Conference and Symposium on Logic Programming*, pages 828–842, Seattle, Washington, August 15–19, 1988.
- [Min82] J. Minker. On Indefinite Databases and the Closed World Assumption. In *Lecture Notes in Computer Science 138*, pages 292–308. Springer-Verlag, 1982.
- [MR88] J. Minker and A. Rajasekar. Procedural Interpretation of Non-Horn Logic Programs. In E. Lusk and R. Overbeek, editors, *Proc. 9th International Conference on Automated Deduction*, pages 278–293, Argonne, IL, May 23–26, 1988.
- [MR90] J. Minker and A. Rajasekar. A Fixpoint Semantics for Disjunctive Logic Programs, 1990. To appear in *Journal of Logic Programming*.
- [MRL] J. Minker, A. Rajasekar, and J. Lobo. Theory of Disjunctive Logic Programs. Submitted.
- [MZ82] J. Minker and G. Zanon. An Extension to Linear Resolution with Selection Function. *Information Processing Letters*, 14(3):191–194, June 1982.
- [PP88] H. Przymusinska and T. Przymusinski. Weakly Perfect Model Semantics for Logic Programs. In R.A. Kowalski and K.A. Bowen, editors, *Proc. 5th International Conference and Symposium on Logic Programming*, pages 1106–1120, Seattle, Washington, August 15–19, 1988.

- [Prz] T. Przymusiński. Stationary semantics for disjunctive logic programs. In preparation.
- [Prz88a] T. Przymusiński. On the Relationship Between Logic Programming and Non-Monotonic Reasoning. In *Proc. AAAI-88*, pages 444–448, 1988.
- [Prz88b] T.C. Przymusiński. Non-monotonic Reasoning vs. Logic Programming – A New Perspective. In Y. Wilks and D. Partridge, editors, *Handbook on the Formal Foundations of AI*. 1988.
- [Prz88c] T.C. Przymusiński. On the Declarative Semantics of Deductive Databases and Logic Programming. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann Pub., Washington, D.C., 1988.
- [Prz89a] T. Przymusiński. Three-valued Formalizations of Non-monotonic Reasoning and Logic programming. In *Proceedings of First International Conference on Knowledge Representation and Reasoning*, 1989.
- [Prz89b] T.C. Przymusiński. Every Logic Program has a Natural Stratification and an Iterated Fixed Point Model. In "*Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principle of Database Systems*", pages 11–21, 1989.
- [RM88] A. Rajasekar and J. Minker. On Stratified Disjunctive Programs. Technical Report CS-TR-2168 UMIACS-TR-88-99, Department of Computer Science, University of Maryland, College Park, December 1988. To appear in the *Journal of Logic Programming*.
- [Ros] K. Ross. Well-founded semantics for disjunctive logic programs. to appear in *International Conference on Deductive and Object Oriented Databases*, Kyoto.
- [She88] J.C. Shepherdson. Negation in Logic Programming. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 19–88. Morgan Kaufman Pub., 1988.

- [VRS88] A. Van Gelder, K. Ross, and J.S. Schlipf. Unfounded Sets and Well-founded Semantics for General Logic Programs. In *Proc. 7th Symposium on Principles of Database Systems*, pages 221–230, 1988.
- [YH85] A. Yahya and L.J. Henschen. Deduction in Non-Horn Databases. *J. Automated Reasoning*, 1(2):141–160, 1985.